Fast Algorithms for the Hypercomplex Fourier Transforms*

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

Abstract

In multi-dimensional signal processing the Clifford Fourier transform (CFT or in the 2-D case: quaternionic Fourier transform/QFT) is a consequent extension of the complex valued Fourier transform. Hence, we need a fast algorithm in order to compute the transform in practical applications. Since the CFT is based on a corresponding Clifford algebra (CA) and CAs are not commutative in general, we cannot simply apply the n-dimensional decimation method as in the complex case. We propose a solution for this problem for real signals. The idea is to embed the CFT in a different algebra which is isomorphic to the m-fold Cartesian product of the complex numbers. Since this approach only works for real signals, we have to develop a different method for Clifford valued signals. We present two ways of calculating the CFT: one using the just mentioned transform for real signals and one approach using 1-D FFTs similar to row-column transforms in the 2-D case. All described algorithms are explicitly formulated for the 3-D case (8-D CA) and the asymptotic complexities are calculated. On modern computers all floating point operations except for divisions and square-roots are performed in the same time. Therefore, we always consider the whole number of floating point operations.

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

1 Introduction and Motivation

The signal theory which is applied in image processing up to now is a simple 'blow-up' extension of the classical one-dimensional signal theory. Since the complex valued signal theory includes several drawbacks for dimensions higher than two, there is a need for an intrinsic n-D signal theory.

- In the n-D complex signal theory Hermite symmetry is lost ($n > 1$). That means that we have no simple symmetry property of the spectrum of a real signal any more.
- The commonly used analytic signal has negative frequencies. Since the spectrum does not contain enough redundancy, we cannot describe the signal with positive frequencies only.
- The phase concept is intrinsically one-dimensional. A complex number cannot have more than one phase. For multi-dimensional signal processing one phase is not sufficient.

These signal theoretic facts yield several practical drawbacks.

- There is no linear approach for the detection of junctions and corners which is comparable to the detection of lines and edges.
- Some distortion only appear for higher dimensions, e.g., rotations. The complex signal theory does not include powerful concepts for dealing with these new distortions.

- The filter design is far from being isotropic. The response of a filter is closely related to its orientation.

Our idea is to develop a hypercomplex signal theory which solves the mentioned problems [5, 16]. We hope that an extended algebraic framework yields more powerful tools for multi-dimensional signal processing. In order to make experiments we need fast algorithms for working with the hypercomplex Fourier transforms. These algorithms will be presented in the following.

2 Basics

In this section we present some algebraic framework about quaternions and Clifford algebras (see e.g. [15, 17]). Furthermore, we define the quaternionic Fourier transform and the Clifford Fourier transform.

2.1 Algebraic Framework

The algebra of quaternions which is denoted by $\mathbb{H}$ is formed by three imaginary units i, j, k where

$$i^2 = j^2 = k^2 = -1 \tag{1}$$

$$ij = -ji = k \quad . \tag{2}$$

These formulae induce the quaternionic product. Addition and difference is defined component-wise. Therefore, the quaternions form an associative but non-commutative algebra (a so called *skew field*). A general quaternion $q = a + bi + cj + dk$ consists of four independent coefficients.

The generalization of complex numbers and quaternions is the Clifford Algebra. In our case, a Clifford Algebra of dimension 2^n is denoted by $\mathbb{R}_{0,n}$. That means that we have an underlying n-D vectorspace with Euclidean norm[1]. From the n basis elements we can construct $2^n - 1$ imaginary units where

- the n basis elements $\mathbf{e}_1, \ldots, \mathbf{e}_n$ square to -1
- we have $\binom{n}{2}$ imaginary units of the form $\mathbf{e}_a\mathbf{e}_b = -\mathbf{e}_b\mathbf{e}_a = \mathbf{e}_{ab}$
- we have $\binom{n}{3}$ imaginary units of the form $\mathbf{e}_a\mathbf{e}_{bc} = \mathbf{e}_{abc}$
- up to one unit $\mathbf{e}_{1\ldots n}$.

Therefore, a general $\mathbb{R}_{0,n}$ element is a 2^n-D vector. The product on $\mathbb{R}_{0,n}$ is defined by the relations between the basis elements and the distributive law.

2.2 Definition of the QFT/CFT

Based on the just introduced algebraic framework, we can define the quaternionic Fourier transform.

Definition 1 (QFT) *The Quaternionic Fourier Transform (QFT) is defined by*

$$F^q(u,v) = \int_{\mathbb{R}^2} e^{-i2\pi ux} f(x,y) e^{-j2\pi vy} dx\, dy \tag{3}$$

This definition can be found in [4, 10, 11].

The QFT is quaternionic Hermite symmetric. That means that the quaternionic spectrum of a real signal has a real part which is symmetric wrt. both coordinate axes and imaginary parts which are symmetric/antisymmetric wrt. the coordinate axes:

	wrt. x-axis	wrt. y-axis
real part	symmetric	symmetric
i-part	antisymmetric	symmetric
j-part	symmetric	antisymmetric
k-part	antisymmetric	antisymmetric

Table 1: Hermite symmetry of the QFT

The generalization of the QFT is the Clifford Fourier transform.

Definition 2 (CFT) *The Clifford Fourier Transform (CFT) is defined by*

$$F^c(\boldsymbol{u}) = \int_{\mathbb{R}^n} f(\boldsymbol{x}) \prod_{j=1}^{n} e^{-\mathbf{e}_j 2\pi u_j x_j} d^n\boldsymbol{x} \tag{4}$$

This definition can be found in [4, 3].

The CFT is Clifford Hermite symmetric, which is the systematic extension of the quaternionic Hermite symmetry. In particular, the spectrum of a real signal is completely contained in one orthant.

[1] Note that the signature of a basis element of a Clifford algebra is the inverse of the quadratic form of the corresponding vectorspace element, see [14].

2.3 Fast Algorithms by DOT

The decimation of time method (for a 1-D signal) is based on two effects. Firstly, the discrete Fourier transform of a signal does not change, if zeroes are inserted between the signal values (see Fig. 1).

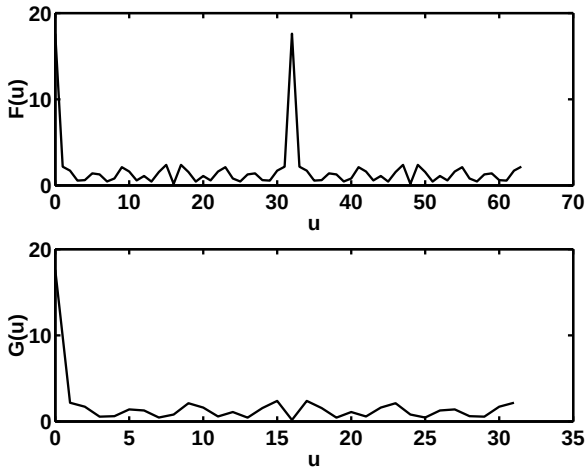

Figure 1: Spectra of a signal with and without inserted zeroes

Secondly, we need the shift theorem of the Fourier transform. For a shift of one position in spatial domain we obtain the phase factor $e^{-i2\pi u}$ in frequency domain.

Using this two effects, we can decompose the 1-D signal f into two signals f_e and f_o (see Fig. 2). The zeroes of f_e and of the shifted version of f_o can be omitted without changing the spectra. Therefore, we obtain the spectrum of f by adding the spectra of the partial signals, where the second spectrum must be multiplied by $e^{i2\pi u}$ in order to eliminate the shift operation.

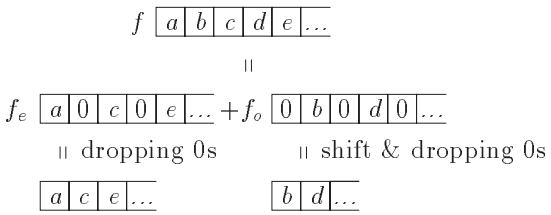

Figure 2: Decimation of time

If we decompose a two-dimensional spectrum, we obtain four addends, where one is unshifted, two are shifted wrt. one coordinate and one is shifted wrt. both coordinates. Together with the re-ordering at the beginning[2], we obtain the following data-flow graph:

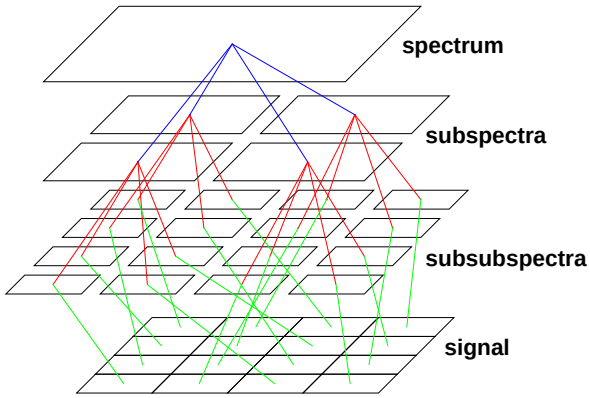

Figure 3: Data-flow in a 2-D decomposition

3 Fast Hypercomplex Transforms

In this section, we firstly apply the DOT-method to the QFT and we show why we cannot apply it to the CFT3. In order to solve this problem, we introduce a new algebra and a new hypercomplex Fourier transform for which we develop fast algorithms in the last part of this section.

3.1 FQFT but no FCFT3

Firstly, we develop a fast quaternionic Fourier transform. The starting point for the fast algorithm is the discrete QFT.

Definition 3 (DQFT) *The discrete quaternionic Fourier transform (DQFT) is defined by*

$$F_{u,v} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e^{-i2\pi ux/N} f_{x,y} e^{-j2\pi vy/N} \quad . \qquad (5)$$

[2]By applying the bit-change-rule, we can omit that the data must be copied during the execution of the FFT algorithm.

By applying the 2-D DOT-method to the DQFT, we obtain the following derivation

$$\begin{aligned} F_{u,v} &= \sum_{x_1,y_1=0}^{N/2-1} \sum_{x_0,y_0=0}^{1} \Big(e^{-i2\pi u(2x_1+x_0)/N} \cdot \\ &\quad \cdot f_{2x_1+x_0,2y_1+y_0} e^{-j2\pi v(2y_1+y_0)/N} \Big) \\ &= F^{ee}_{u,v} + e^{-i2\pi u/N} F^{oe}_{u,v} + F^{eo}_{u,v} e^{-j2\pi v/N} + \\ &\quad + e^{-i2\pi u/N} F^{oo}_{u,v} e^{-j2\pi v/N} \quad . \end{aligned} \tag{6}$$

In order to understand the problem which occurs in the context of fast algorithms in Clifford algebra, we can reduce our considerations to the shift theorem. The decomposition of the spectrum does not yield any problem if the underlying algebra is changed. Therefore, we only have to explore the effect of shift operations in the spectrum.

For the QFT, we have the following shift theorem:

$$f(x+\xi, y+\eta) \circ\!\!-\!\!\bullet\; e^{-i2\pi u\xi} F^q(u,v) e^{-j2\pi v\eta} \quad . \tag{7}$$

Therefore, we obtain (6).

The question is now if we can state such a factorized shift theorem for higher dimensions, too:

$$\begin{aligned} &f(x+\xi, y+\eta, z+\omega) \;\overset{?}{\circ\!\!-\!\!\bullet}\; \\ &F^c(u,v,w) e^{-\mathbf{e}_1 2\pi u\xi} e^{-\mathbf{e}_2 2\pi v\eta} e^{-\mathbf{e}_3 2\pi w\omega} \end{aligned} \quad ? \tag{8}$$

The answer is obviously *no*, due to the non-commutativity of Clifford algebras. We have two ways of multiplication, this is from the left and from the right. Therefore, we can circumvent the problem of the non-commutativity in the quaternion algebra, i.e. we can factorize terms of $\mathbf{e}_1$ to the left and terms of $\mathbf{e}_2$ to the right:

$$e^{\mathbf{e}_1(\phi_1+\phi_2)} e^{\mathbf{e}_2(\phi_3+\phi_4)} = e^{\mathbf{e}_1\phi_1} \big(e^{\mathbf{e}_1\phi_2} e^{\mathbf{e}_2\phi_4} \big) e^{\mathbf{e}_2\phi_3} \quad .$$

But we cannot factorize the CFT kernel if we have more than two different imaginary units in the exponential functions because at least two terms have to commute:

$$e^{\mathbf{e}_1\phi_1} e^{\mathbf{e}_2(\phi_2+\theta)} e^{\mathbf{e}_3\phi_3} \neq e^{\mathbf{e}_1\phi_1} e^{\mathbf{e}_2\phi_2} e^{\mathbf{e}_3\phi_3} e^{\mathbf{e}_2\theta} \quad .$$

Therefore, we cannot apply the DOT-method for CFTn with $n > 2$.

3.2 Commutative Hypercomplex Fourier Transform and its Properties

The basic idea to overcome this problem is to embed the hypercomplex Fourier transform in a slightly different algebra. This new algebra has to be commutative. We reach this aim by changing the rules for the multiplication (see [9]):

$$\mathbf{e}_1{}'\mathbf{e}_2{}' = \mathbf{e}_2{}'\mathbf{e}_1{}' = \mathbf{e}_{12}{}' \quad . \tag{9}$$

Obviously, we create a commutative hypercomplex algebra (see [21]) by these rules. Analogously to Clifford algebras, we denote the commutative algebras by $\mathcal{H}_n$. By the mapping

$$\begin{aligned} \mathbf{e}_1{}' &\mapsto i \otimes 1 \otimes 1 \otimes \dots \\ \mathbf{e}_2{}' &\mapsto 1 \otimes i \otimes 1 \otimes \dots \\ &\vdots \end{aligned}$$

we obtain an isomorphism between $\mathcal{H}_n$ and $\mathbb{C}^{\otimes n}$.

In the algebra $\mathcal{H}_n$ we can define an integral transform which is similar to the CFT:

Definition 4 (HFT) *The commutative hypercomplex Fourier transform (HFT) is defined by*

$$F^h(\boldsymbol{u}) = \int_{\mathbb{R}^n} f(\boldsymbol{x}) e^{-2\pi \sum_{j=1}^{n} \mathbf{e}_j{}' u_j x_j} d^n\boldsymbol{x} \tag{10}$$

This definition can be found in [12].

The HFT is hypercomplex Hermite symmetric which means that it has the same symmetry properties as the CFT.

Actually, there are even more similarities to the CFT. If we consider the CFT of a real signal, the CFT only consists of *ordered* multiplications (ordered wrt. the index set of the imaginary units). Therefore, the HFT and the CFT yield the same coefficients and by a simple exchange of the imaginary units with the same index we can obtain one transform from the other one.

Furthermore, we have proved that the n-fold tensor product of the algebra of complex numbers is isomorphic to the 2^{n-1}-fold Cartesian product of complex numbers:

$$\mathbb{C}^{\otimes n} \cong \mathbb{C}^{2^{n-1}} \quad , \tag{11}$$

see, for example, [13]. The proof makes use of the matrix representation of both algebras. In particular, we perform an eigenvalue transform on the matrix representation of $\mathcal{H}_n$.

The eigenvectors of the eigenvalue transform yield a correspondence of the HFTn and the complex FTn, where $i = \mathbf{e}_1{}'$:

$$F^h(u,v) = \frac{(1-\mathbf{e}_{12}{}')F(u,v) + (1+\mathbf{e}_{12}{}')F(u,-v)}{2}$$

for $n = 2$ and

$$\begin{aligned} F^h(u,v,w) \\ &= \big(F(u,v,w)(1-\mathbf{e}_{12}{}'-\mathbf{e}_{13}{}'-\mathbf{e}_{23}{}') \\ &+ F(u,v,-w)(1-\mathbf{e}_{12}{}'+\mathbf{e}_{13}{}'+\mathbf{e}_{23}{}') \\ &+ F(u,-v,w)(1+\mathbf{e}_{12}{}'-\mathbf{e}_{13}{}'+\mathbf{e}_{23}{}') \\ &+ F(u,-v,-w)(1+\mathbf{e}_{12}{}'+\mathbf{e}_{13}{}'-\mathbf{e}_{23}{}'))/4 \end{aligned}$$

for $n = 3$.

3.3 Two Approaches for FAs

Using the HFT, we are able to introduce a straightforward fast algorithm, since the shift theorem of the HFT reads

$$\begin{aligned} &f(x+\xi, y+\eta, z+\omega) \\ &\quad\Big\downarrow \\ &F^h(u,v,w)e^{-2\pi(\mathbf{e}_1{}'u\xi+\mathbf{e}_2{}'v\eta+\mathbf{e}_3{}'w\omega)} \quad . \end{aligned} \tag{12}$$

Therefore, we can apply the decimation of time method. For the case $n = 3$ we obtain (using the abbreviations: $w^{\mathbf{e}_1{}'} = e^{-2\pi\mathbf{e}_1{}'u}$ etc.)

$$\begin{aligned} F^h = F^h_{eee} + F^h_{oee}w^{\mathbf{e}_1{}'} + F^h_{eoe}w^{\mathbf{e}_2{}'} \\ + F^h_{ooe}w^{\mathbf{e}_1{}'}w^{\mathbf{e}_2{}'} + F^h_{eeo}w^{\mathbf{e}_3{}'} + F^h_{oeo}w^{\mathbf{e}_1{}'}w^{\mathbf{e}_3{}'} \\ + F^h_{eoo}w^{\mathbf{e}_2{}'}w^{\mathbf{e}_3{}'} + F^h_{ooo}w^{\mathbf{e}_1{}'}w^{\mathbf{e}_2{}'}w^{\mathbf{e}_3{}'} \quad , \end{aligned}$$

which yields a fast algorithm for real and $\mathcal{H}_3$-valued 3-D signals.

If we want to transform $\mathbb{R}_{0,3}$-valued signals, we can develop a fast algorithm by decomposing the signal into 8 real signals $f_0, f_1, f_2, f_{12}, f_3, f_{13}, f_{23}, f_{123}$ and calculating eight HFTs afterwards. Due to linearity we obtain the CFT from the HFTs by:

$$\begin{aligned} F^c = F^h_0 + \mathbf{e}_1F^h_1 + \mathbf{e}_2F^h_2 + \mathbf{e}_{12}F^h_{12} + \mathbf{e}_3F^h_3 \\ + \mathbf{e}_{13}F^h_{13} + \mathbf{e}_{23}F^h_{23} + \mathbf{e}_{123}F^h_{123} \quad , \end{aligned}$$

where $\mathbf{e}_.{}'$ is substituted by $\mathbf{e}_.$.

A second idea for developing fast algorithms is to use complex FFT2 (FFT3) algorithms and to apply the isomorphism between $\mathcal{H}_2$ and $\mathbb{C}^2$ ($\mathcal{H}_3$ and $\mathbb{C}^4$). This method works for real and $\mathcal{H}_2$- ($\mathcal{H}_3$-)valued signals.

If we consider that the most FFT2/FFT3 algorithms are implemented as a *row-column* algorithm of the FFT1 (i.e. the separability of the kernel is utilized, see [2, 18]) we can optimize the second idea. We can *directly* apply the row-column method to the HFT (i.e. separate the HFT kernel) and obtain algorithms which are based on complex FFT1s. The following table shows the number of FFT1s needed for the HFT2 and the HFT3, respectively:

	HFT2	HFT3
real	3	7
$\mathcal{H}_{2/3}$-valued	4	12

Table 2: Number of FFT1s

Now, the question rises if we can use a similar construction for calculating the QFT. If we separate the QFT kernel, we obtain

$$\begin{aligned} w^i f w^j \quad = \quad & \big(w^i(f_r + if_i) + w^i(f_j + if_k)j\big)w^j \\ & \quad Re\ Im \qquad\quad Re \quad Im \\ & (F_r + jF_j)w^j + i(F_i + jF_k)w^j \end{aligned}$$

which is not so surprising because we can directly develop a fast algorithm for the QFT by decimation of time. The question which is more interesting is if we can apply the analogous construction for the CFT3. The CFT3 kernel itself is separable, of course, but the missing commutativity impedes the factorization of the whole transform. Nevertheless, by changing some signs, we obtain the following set of equations, which yields a row-column algorithm for the CFT3 with the same number of FFT1s as the algorithm for the HFT3.

$$
\begin{aligned}
&(f_0 + f_1\mathbf{e}_1 + f_2\mathbf{e}_2 + f_{12}\mathbf{e}_{12} + f_3\mathbf{e}_3 + f_{13}\mathbf{e}_{13} + f_{23}\mathbf{e}_{23} + f_{123}\mathbf{e}_{123})w^{\mathbf{e}_1}w^{\mathbf{e}_2}w^{\mathbf{e}_3} \\
&= (\underbrace{(f_0 + f_1\mathbf{e}_1)w^{\mathbf{e}_1}}_{\searrow} + \underbrace{\mathbf{e}_2(f_2 - f_{12}\mathbf{e}_1)w^{\mathbf{e}_1}}_{\downarrow} + \underbrace{\mathbf{e}_3(f_3 - f_{13}\mathbf{e}_1)w^{\mathbf{e}_1}}_{\swarrow} + \underbrace{\mathbf{e}_{23}(f_{23} + f_{123}\mathbf{e}_1)w^{\mathbf{e}_1}}_{\swarrow})w^{\mathbf{e}_2}w^{\mathbf{e}_3} \\
&= (\overbrace{\hat{f}_0 + \hat{f}_1\mathbf{e}_1}^{\searrow} + \overbrace{\hat{f}_2\mathbf{e}_2 - \hat{f}_{12}\mathbf{e}_{12}}^{\downarrow} + \overbrace{\hat{f}_3\mathbf{e}_3 - \hat{f}_{13}\mathbf{e}_{13}}^{\swarrow} + \overbrace{\hat{f}_{23}\mathbf{e}_{23} + \hat{f}_{123}\mathbf{e}_{123}}^{\swarrow})w^{\mathbf{e}_2}w^{\mathbf{e}_3} \\
&= (\underbrace{(\hat{f}_0 + \hat{f}_2\mathbf{e}_2)w^{\mathbf{e}_2}}_{\searrow} + \underbrace{\mathbf{e}_1(\hat{f}_1 - \hat{f}_{12}\mathbf{e}_2)w^{\mathbf{e}_2}}_{\downarrow} + \underbrace{\mathbf{e}_3(\hat{f}_3 - \hat{f}_{23}\mathbf{e}_2)w^{\mathbf{e}_2}}_{\swarrow} - \underbrace{\mathbf{e}_{13}(\hat{f}_{13} + \hat{f}_{123}\mathbf{e}_2)w^{\mathbf{e}_2}}_{\swarrow})w^{\mathbf{e}_3} \\
&= (\overbrace{F_0 + F_2\mathbf{e}_2}^{\searrow} + \overbrace{F_1\mathbf{e}_1 + F_{12}\mathbf{e}_{12}}^{\downarrow} + \overbrace{F_3\mathbf{e}_3 - F_{23}\mathbf{e}_{23}}^{\swarrow} - \overbrace{F_{13}\mathbf{e}_{13} + F_{123}\mathbf{e}_{123}}^{\swarrow})w^{\mathbf{e}_3} \\
&= (F_0 + F_3\mathbf{e}_3)w^{\mathbf{e}_3} + \mathbf{e}_2(F_2 - F_{23}\mathbf{e}_3)w^{\mathbf{e}_3} + \mathbf{e}_1(F_1 - F_{13}\mathbf{e}_3)w^{\mathbf{e}_3} + \mathbf{e}_{12}(F_{12} + F_{123}\mathbf{e}_3)w^{\mathbf{e}_3}
\end{aligned}
$$

4 Complexities

We have presented so far different algorithms for calculating the 2-D and 3-D hypercomplex transforms of a real or hypercomplex signal. In the following table the complexities of all considered algorithms can be found ($k = \log_2 N$). Since on modern computers, all floating point operations except for divisions and square-roots are performed in the same time (see [22, 20, 8]), we have only considered the total number of operations. Furthermore, we have omitted the possibility of calculating the hypercomplex spectra of a real signal from the Hartley transform of the signal.

algorithm	operations
FQFT/FHFT2 real*	$6kN^2$
FQFT/FHFT2 hypercomplex	$24kN^2$
2-D row-column real**	$7.5kN^2$
2-D row-column hypercomplex	$20kN^2$
FHFT3/FCFT3 real*	$11.5kN^3$
FHFT3 hypercomplex	$92kN^3$
3-D row-column real**	$17.5kN^3$
3-D row-column hypercomplex	$60kN^3$
FCFT3 by 8 FHFT3 (real)	$92kN^3$

*optimized real transforms (e.g. overlapping [6, 7])
**FFT1 with $2.5kN$ operations (optimized)

Table 3: Complexities of the considered algorithms

The following diagrams visualize the results from table 3. Note that we did not separate the sign-permutation matrix for the 2-D DOT algorithms. This would lower the arithmetic complexity of the FQFT to $5kN^2$. In comparison, the FFT2 (by 2-D DOT) has a complexity of $6.25kN^2$ ($4.25kN^2$ with separation) arithmetic operations. Therefore, the graph for the FFT2 is (nearly) the same as the one for the FQFT in Fig. 4.

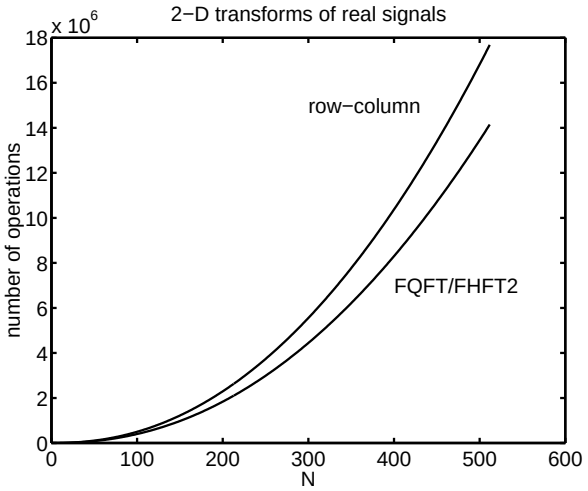

Figure 4: Complexities for transforms of real 2-D signals

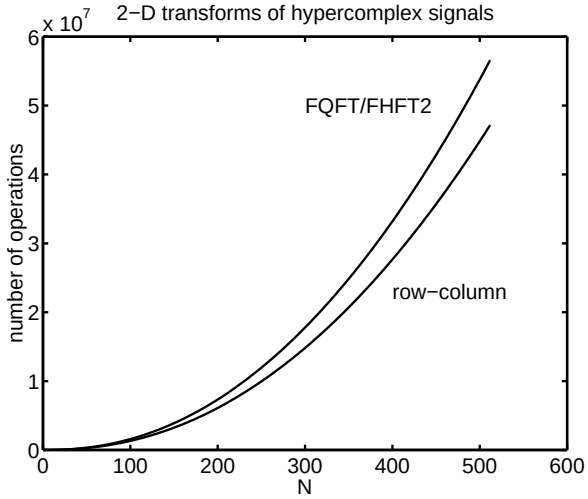

Figure 5: Complexities for transforms of hypercomplex 2-D signals

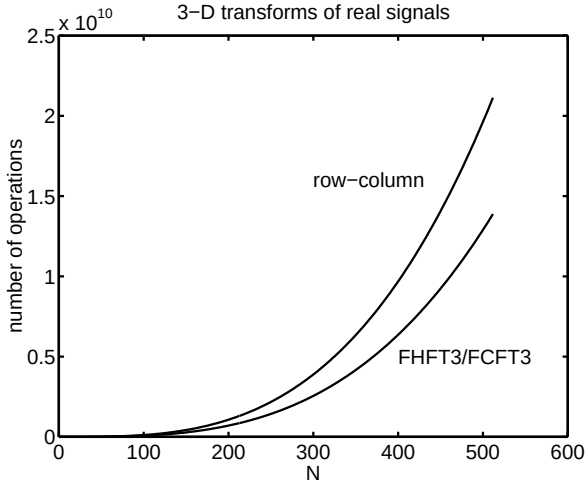

Figure 6: Complexities for transforms of real 3-D signals

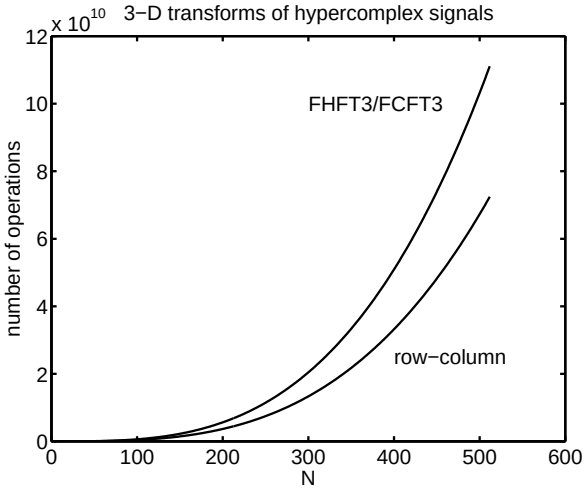

Figure 7: Complexities for transforms of hypercomplex 3-D signals

5 Conclusion and Discussion

We can summarize the results of this paper as follows:

- The CFT is not the only transform which separates all symmetries. Therefore, the HFT has a right to exist besides the CFT.
- The HFT is graphically even more intuitive since the algebra $\mathcal{H}_n$ can be interpreted as the tensor product of n complex planes.
- We have shown that the n-D FCFT/FHFT algorithms are as fast as the n-D FFT algorithms (for real signals).
- There is a linear relation between the complexity increase and the dimension of the algebra.
- The results in the table of complexities and the corresponding diagrams show that for hypercomplex signals the row-column method is superior to the decimation of time method.
- The implementation of the row-column method is faster for real signals, too, though its asymptotic complexity is higher.

In this paper, we have presented the algorithmic framework which is needed for further research on hypercomplex signal analysis. Especially the design of hypercomplex filters will be an interesting object in future. We will try to find new signal embeddings in order to obtain more powerful spectral representations (see e.g. [1, 19]). For all these approaches and the accompanying experiments, the existence of fast algorithms is of fundamental importance.

References

[1] T. Bülow. *Global and Local Hypercomplex Spectral Signal Representations for Image Processing and Analysis*. PhD thesis, Christian-Albrechts-University of Kiel, 1999.

[2] R. N. Bracewell. *The Fourier transform and its applications*. McGraw Hill, 1986.

[3] F. Brackx, R. Delanghe, and F. Sommen. *Clifford Analysis*. Pitman, Boston, 1982.

[4] 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.

[5] 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, AFPAC'97, Kiel*, volume 1315 of *LNCS*, pages 148–163. Springer, 1997.

[6] 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.

[7] M. A. Chichyeva and M. V. Pershina. On Various Schemes of 2D-DFT Decomposition with Data Representation in the Quaternion Algebra. *Image Processing & Communications*, 2(1):13–20, 1997.

[8] Digital Equipment Corporation. Digital Semiconductor Alpha 21164PC Microprocessor Data Sheet[3], 1997.

[9] Clyde M. Davenport. A Commutative Hypercomplex Algebra with Associated Function Theory. In R. Ablamowicz, editor, *Clifford Algebras with Numeric and Symbolic Computations*, pages 213–227. Birkhäuser Bosten, 1996.

[10] T. A. Ell. *Hypercomplex Spectral Transformations*. PhD thesis, University of Minnesota, 1992.

[11] R.R. Ernst, W.P. Aue, P. Bachmann, J. Karhan, A . Kumar, and L. Müller. Two-dimensional NMR spectroscopy. In *Proc. 4th Ampère Int. Summer School, Pula, Yugoslavia*, 1976.

[12] M. Felsberg. Signal Processing Using Frequency Domain Methods in Clifford Algebra[4]. Master's thesis, Christian-Albrechts-University of Kiel, 1998.

[13] M. Felsberg et al. Commutative Hypercomplex Fourier Transforms of Multidimensional Signals. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, Springer Series in Information Sciences. Springer, Berlin, 1999. to appear.

[14] J. E. Gilbert and M. A. M. Murray. *Clifford algebras and Dirac operators in harmonic analysis*. Cambridge University Press, 1991.

[15] D. Hestenes and G. Sobczyk. *Clifford algebra to geometric calculus, A Unified Language for Mathematics and Physics*. Reidel, Dordrecht, 1984.

[16] B. Jähne, J. Haußecker, and P. Geißler. *Handbook of Computer Vision and Applications*. Academic Press, Boston, 1999.

[17] I. R. Porteous. *Clifford Algebras and the Classical Groups*. Cambridge University Press, 1995.

[18] W. Press et al. *Numerical Recipes in C*. Cambridge University Press, 1994.

[19] E. Rundblad and V. Labunets. Is the Brain a "Clifford Algebra Computer"? In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, Springer Series in Information Sciences. Springer, Berlin, 1999. to appear.

[20] Silicon Graphics, Inc. MIPS RISC Technology R10000 Microprocessor Technical Brief[5], 1998.

[21] I. L. Kantor & A. S. Solodovnikov. *Hypercomplex Numbers*. Springer Verlag, New-York, 1989.

[22] Sun Microsystems, Inc. UltraSPARC-II Data Sheet[6], 1998.

[3] http://ftp.digital.com/pub/DECinfo/semiconductor/literature/164pcds.pdf

[4] http://www.ks.informatik.uni-kiel.de/~mfe/research.html

[5] http://www.sgi.com/processors/r10k/tech_info/Tech_Brief.html

[6] http://www.sun.com/microelectronics/datasheets/stp1031/index2.html