数理解析研究所講究録 1378 巻 2004 年 51-69

Applications of Geometric Algebra in Robot Vision

University of Kiel, Cognitive Systems Group
Gerald Sommer

Abstract

In this tutorial paper we will report on our experience in the use of geometric algebra (GA) in robot vision. The results could be reached in a long term research programme on modelling the perception-action cycle within geometric algebra. We will pick up three important applications from image processing, pattern recognition and computer vision. By presenting the problems and their solutions from an engineering point of view, the intention is to stimulate other applications of GA.

1 Introduction

In this paper we want to present a tutorial survey on applications of geometric algebra (GA) in robot vision. We will restrict our scope to results contributed by the Kiel Cognitive Systems Group in the last few years. For more details and for getting a wider view on this topic a visit of the publications on the website http://www.ks.informatik.uni-kiel.de is recommended. Especially the tutorial reports [14] and [17] may be helpful.
We will take on an engineer's viewpoint to give some impression of the need of such complex mathematical framework as geometric algebra for designing robot vision systems more easily. In fact, we are using GA as a mathematical language for modelling. This includes the task related shaping of that language itself.

1.1 Robot Vision

The aim of robot vision is to make robots seeing, i.e. to endow robots with visual capabilities comparable to those of human. While contemporary applications of robots are restricted to industrial artificial environments, the hope is that future generations of robots are able to cope with real world conditions. The term robot vision indicates a concentration of research activities onto modelling of useful visual architectures. On the other hand the term visual robotics emphasizes the control of robots by visual sensoric information.

Obviously, both aspects are very tightly related. In the framework of behaviour-based robotics this coupling of (visual) perception and action is the key paradigm of system design. A behaviour is represented in the so-called perception-action cycle. Of practical importance are the projections of the perception-action cycle: "vision for action" means controlling actions by vision and "action for vision" means controlling the gaze (or body) for making vision more easier.
Robot vision is emerging from several contributing disciplines which are as different as image processing, computer vision, pattern recognition and robotics. Each of these have their own scientific history and are using different mathematical languages. The demanding task of system design is aiming at a unique framework of modelling. With respect to the quest of a useful cognitive architecture the perception-action cycle may be the right representation. With respect to unifying the mathematical language geometric algebra is hopefully the framework of merging the above mentioned disciplines in a coherent system.

1.2 Motivating Geometric Algebra

Such system has to be an embodiment of the geometry and the stochastic nature of the external world. This should be understood in a dynamic sense which enables internal processes converging at reasonable interpretations of the world and useful actions in the environment.
While merging the different disciplines is one main motivation of using geometric algebra in robot vision, the other one is to overcome several serious limitations of modelling within the disciplines themselves. These limitations result from the dominant use of vector algebra. A vector space is a completely unstructured algebraic framework endowed with a simple product, the scalar product, which only can destroy information originally represented in the pair of vectors by mapping them to a scalar.
Imagine, for instance, that a cognitive system as a human could reason on the world or could act in the world only by its decomposition into a set of points and having no other operations at hand than the scalar product. This in fact is an impossible scenario. The phenomena of the world we can cope with are of global nature at the end in comparison to the local point-like entities we have in vector space. They are phenomena of higher order in the language of vector algebra. Hence, most of the basic disciplines of robot vision are getting stuck in non-linearities because of the non-tractable complexity of the problem at hand.
In fact, only the tight relations of GA to geometric modelling [12] have been considered, yet. This is what we want to review here. The benefit we derive from using GA is rooted in the rich algebraic structure of the 2^n-dimensional linear space $\mathbb{R}_{p,q,r}$ resulting from the vector space $\mathbb{R}^{p,q,r}$. The indexes p, q, r with $p + q + r = n$ mark its signature. By choosing a certain signature the decision for certain geometries adequate to the problem at hand can be made. On the other hand, the blade structure of a GA is representing higher-order relations between vectors which, once computed, can be further processed in a linear manner. Hence, multi-dimensional or higher-order phenomena can be reduced

to one-dimensional or linear representations. This has not only impact on modelling of geometric problems but of stochastic ones also. Therefore, there is need of using GA in context of higher-order statistics too.
In the following sections we want to show the advantages of using GA in three different areas of robot vision. These are signal analysis, pattern recognition using the paradigm of neural computing, and computer vision. In each case we will emphasize the inherent limitations of modelling in the classical scheme and the benefits we gain from using GA instead. In the conclusion section we will give a summary of all these advantages in a more general way.

2 Analysis of Multi-dimensional Signals

Image analysis as a fundamental part of robot vision does not mean to deliver complete descriptions of scenes or to recognize certain objects. This in fact is subject of computer vision. Instead, the aim of image analysis is deriving a set of rich features from visual data for further processing and analysis. In (linear) signal theory we are designing (linear shift invariant) operators which should extract certain useful features from signals. The Fourier transform plays a major role because representations of both operators and images in spectral domain are useful for interpretation.
In this section we want to give an overview on our endavours of overcoming a not well-known representation gap in Fourier domain in case of multi-dimensional signals. In fact, the well-known complex valued Fourier transform cannot explicitely represent multi-dimensional phase concepts. The Fourier transform is a global integral transform and amplitude and phase are global spectral representations. More important in practice is the incompleteness of local spectral representations in multi-dimensional case. This is a representation problem of the Hilbert transform which delivers the holomorphic complement of a one-dimensional real valued signal. Therefore, the aim of this section is showing generalizations of both Fourier and Hilbert transform from the one-dimensional to the multi-dimensional case which are derived from embeddings into geometric algebra. In that respect two different concepts of the term dimension of a function have to be distinguished. The first one is the global embedding dimension, which is related to the Fourier transform. The second one is the (local) intrinsic dimension, which is related to the Hilbert transform and which means the degrees of freedom that locally define the function.

2.1 Global Spectral Representations

It is well-known that the complex valued Fourier transform, $F^c \in \mathbb{C}$,

$$F^c(\mathbf{u}) = \mathcal{F}^c\{f(\mathbf{x})\} = \int f(\mathbf{x}) \exp(-j2\pi\mathbf{u}\mathbf{x})\, d^N\mathbf{x} \tag{1}$$

enables computing the spectral representations amplitude $A^c(\mathbf{u})$ and phase $\Phi^c(\mathbf{u})$,

$$A^c(\mathbf{u}) = |F^c(\mathbf{u})| \tag{2a}$$

$$\Phi^c(\mathbf{u}) = \arg(F^c(\mathbf{u})), \tag{2b}$$

of the real valued N-dimensional (ND) function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^N$. Although it is commonsense that the phase function is representing the structural information of the function, there is no way in complex domain to explicitly access to ND structural information for $N > 1$. This problem is related to the impossibility of linearly representing symmetries of dimension $N > 1$ in complex domain.
In fact, the Fourier transform in 1D case can be seen as a way to compute the integral over all local symmetry decompositions of the function $f(x)$ into an even and an odd component according to

$$f(x) = f_e(x) + f_o(x) \tag{3}$$

$$F^c(u) = F^c_e(u) + F^c_o(u). \tag{4}$$

This results from the Euler decomposition of the Fourier basis functions

$$Q^c(u,x) = \exp(-j2\pi ux) = \cos(2\pi ux) - j\sin(2\pi ux). \tag{5}$$

In case of 2D complex Fourier transform the Euler decomposition of the basis functions

$$Q^c(\mathbf{u},\mathbf{x}) = \exp(-j2\pi\mathbf{ux}) = \exp(-j2\pi ux)\exp(-j2\pi vy) \tag{6}$$

corresponds to a signal decomposition into products of symmetries with respect to coordinate axes,

$$F^c(\mathbf{u}) = F^c_{ee}(\mathbf{u}) + F^c_{oo}(\mathbf{u}) + F^c_{oe}(\mathbf{u}) + F^c_{eo}(\mathbf{u}). \tag{7}$$

But this concept of line symmetry is partially covered in complex domain because of the limited degrees of freedom. If $F^c_R(\mathbf{u})$ and $F^c_I(\mathbf{u})$ are the real and imaginary components of the spectrum, respectively, $F^c(\mathbf{u}) = F^c_R(\mathbf{u}) + jF^c_I(\mathbf{u})$, then

$$F^c_R(\mathbf{u}) = F^c_{ee}(\mathbf{u}) + F^c_{oo}(\mathbf{u}) \ , \ F^c_I(\mathbf{u}) = F^c_{eo}(\mathbf{u}) + F^c_{oe}(\mathbf{u}). \tag{8}$$

If we consider the Hartley transform [10] as a real valued Fourier transform, $F^r(u)$, then also in 1D case the components $F^r_e(u)$ and $F^r_o(u)$ are totally covered in $F^r \in \mathbb{R}$. This observation supports the following statement [6]. Given a real valued function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^N$, then all its global decompositions into even and odd symmetry components with respect to coordinate axes is given by the Clifford valued Fourier transform $F^{Cl} \in \mathbb{R}_{2^N}$,

$$F^{Cl}(\mathbf{u}) = \mathcal{F}^{Cl}\{f(\mathbf{x})\} = \int_{\mathbb{R}^N} f(\mathbf{x}) \prod_{k=1}^{N} \exp(-i_k 2\pi u_k x_k)\, d^N\mathbf{x}. \tag{9}$$

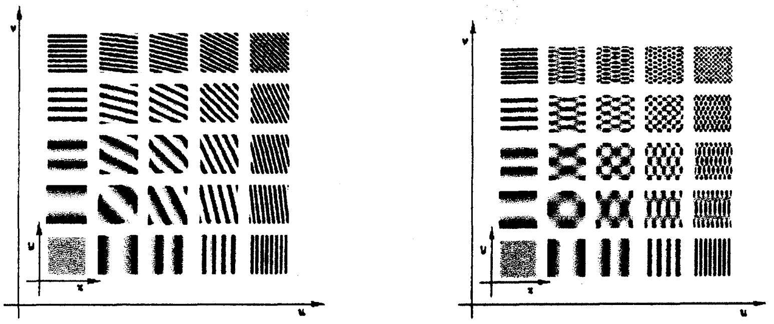

Figure 1: Left: basis functions of the complex 2D Fourier transform. Right: basis functions of the quaternionic 2D Fourier transform. Only the even/even-even part is shown each.

In 2D case the Fourier transform has to be quaternionic, $F^q \in \mathbb{H}$,

$$F^q(\mathbf{u}) = \mathcal{F}^q\{f(\mathbf{x})\} = \int_{\mathbb{R}^2} f(\mathbf{x}) \exp(-i2\pi ux) \exp(-j2\pi vy)\, d^2\mathbf{x} \tag{10}$$

with

$$Q^q(\mathbf{u},\mathbf{x}) = \exp(-i2\pi ux) \exp(-j2\pi vy). \tag{11}$$

Because

$$F^q(\mathbf{u}) = F^q_R(\mathbf{u}) + iF^q_I(\mathbf{u}) + jF^q_J(\mathbf{u}) + kF^q_K(\mathbf{u}) \tag{12a}$$

$$F^q(\mathbf{u}) = F^q_{ee}(\mathbf{u}) + F^q_{oe}(\mathbf{u}) + F^q_{eo}(\mathbf{u}) + F^q_{oo}(\mathbf{u}), \tag{12b}$$

all symmetry combinations are explicitely accessible by one real and three imaginary parts. As figure 1 visualizes, the basis functions according to (11) are intrinsically two-dimensional in contrast to those of (6) and, thus, they can represent 2D structural information. This can also be seen in the polar representation of the quaternionic Fourier transform [6], where ϕ and θ represent each a 1D phase, ψ is a 2D phase.

$$F^q(\mathbf{u}) = |F^q(\mathbf{u})| \exp(i\phi(\mathbf{u})) \exp(k\psi(\mathbf{u})) \exp(j\theta(\mathbf{u})). \tag{13}$$

2.2 Local Spectral Representations

As local spectral representations we call functions of local energy, $e(x)$, (or amplitude, $a(x) = \sqrt{e(x)}$), and local phase, $\phi(x)$,

$$e(x) = f^2(x) + f_H^2(x) \tag{14a}$$

$$\phi(x) = \arg\left(f_A(x)\right), \tag{14b}$$

which are derived from a complex valued extension, $f_A(x)$, of a real 1D function $f(x)$ by an operator $\mathcal{A}$,

$$f_A(x) = \mathcal{A}\{f(x)\} = f(x) + jf_H(x). \tag{15}$$

In signal theory $f_A(x)$ is called analytic signal and $f_H(x)$ is derived from $f(x)$ by a local phase shift of $-\frac{\pi}{2}$, which is gained by the Hilbert transform, $\mathcal{H}$, of $f(x)$,

$$f_H(x) = \mathcal{H}\{f(x)\}. \tag{16}$$

This corrresponds in complex analysis to compute the holomorphic complement of $f(x)$ by solving the Laplace equation. In complex Fourier domain the Hilbert transform reads

$$H(u) = -j\frac{u}{|u|}, \tag{17}$$

thus, the operator of the analytic signal is given by the frequency transfer function

$$A(u) = 1 + \frac{u}{|u|}. \tag{18}$$

The importance of the analytic signal in signal analysis results from the evaluation of the local spectral representations, equations (14a) and (14b): If the local energy exceeds some threshold at x, then this indicates an interesting location. The evaluation of the local phase enables a qualitative signal interpretation with respect to a mapping of the signal to a basis system of even and odd symmetry according to equation (3). For more details see [17]. In image processing we can interprete lines as even symmetric structures and edges as odd symmetric ones, both of intrinsic dimension one. But in images we have as an additional unknown the orientation of lines or edges. Hence, the operators $\mathcal{A}$ or $\mathcal{H}$, respectively, should be rotation invariant. Although the analytic signal is also used in image processing since one decade, only by embedding into GA, this problem could be solved.
Simply to take over the strategy described in section 2.1 from global to local domain is not successful. The quaternionic analytic signal [7], which is derived from a quaternionic Hilbert transform delivers no rotation invariant local energy. Hence, the applied concept of line symmetry cannot succeed in formulating isotropic operators.
Instead, an isotropic generalization of the Hilbert transform in case of a multidimensional embedding of a 1D function can be found by considering point symmetry in the image plane. This generalization is known in Clifford analysis [3] as Riesz transform.
The Riesz transform of a real ND function is an isotropic generalization of the Hilbert transform for $N > 1$ in a $(N+1)$D space. In Clifford analysis, i.e. an extension of the complex analysis of 1D functions to higher dimensions, a real ND function is considered as a harmonic potential field, $\mathbf{f}(\mathbf{x})$, in the geometric algebra $\mathbb{R}_{N+1}$. The Clifford valued extension of $\mathbf{f}(\mathbf{x})$ is called monogenic extension and is located in the hyperplane where

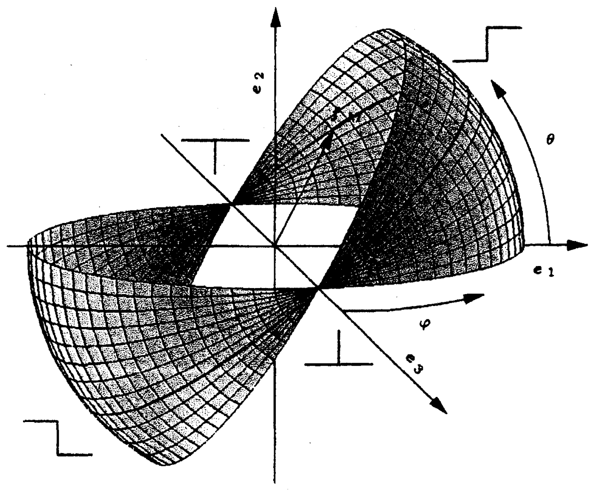

Figure 2: The phase decomposition of a monogenic signal expressing symmetries of a local i1D structure embedded in $\mathbb{R}_3$. The great circle is passing $|f_M| e_3$ and f_M defines the orientation of a complex plane in $\mathbb{R}^3$.

the $(N+1)$th component, say s, vanishes. It can be computed by solving the $(N+1)$D Laplace equation of the corresponding harmonic potential as a boundary value problem of the second kind for $s=0$ (see [8]).
Let us consider the case of a real 2D signal, $f(\mathbf{x})$, represented as $\mathbf{e}_3$-valued vector field $\mathbf{f}(\mathbf{x}) = f(\mathbf{x})\mathbf{e}_3$ in the geometric algebra $\mathbb{R}_3$. Then in the plane $s=0$ a monogenic signal $\mathbf{f}_M$ exists [9],

$$\mathbf{f}_M(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \mathbf{f}_R(\mathbf{x}), \tag{19}$$

where $\mathbf{f}_R(\mathbf{x})$ is the vector of the Riesz transformed signal. In our considered case the monogenic signal is a generalization of the analytic signal, equation (15), for embedding an intrinsically 1D function into a 2D signal domain. Similarly the Riesz transform with the impulse response

$$\mathbf{h}_R(\mathbf{x}) = \frac{\mathbf{x}\mathbf{e}_3}{2\pi|\mathbf{x}|^3} = -\frac{x}{2\pi|\mathbf{x}|^3}\mathbf{e}_{31} + \frac{y}{2\pi|\mathbf{x}|^3}\mathbf{e}_{23} \tag{20}$$

and the frequency transfer function

$$\mathbf{H}_R(\mathbf{u}) = \frac{\mathbf{u}}{|\mathbf{u}|}\mathbf{I}_2^{-1} \quad , \mathbf{I}_2 = \mathbf{e}_{12} \tag{21}$$

generalizes all properties known from the Hilbert transform to 2D.
The local energy, derived from the monogenic signal is rotation invariant. Furthermore, the phase generalizes to a phase vector, whose one component is the normal phase angle and the second component represents the orientation angle of the intrinsic 1D structure in the image plane. Hence, the set of local spectral features is augmented by a feature of

geometric information, that is the orientation. In figure 2 can be seen how the complex plane of the local spectral representations rotates in the augmented 3D space. This results in an elegant scheme of analyzing intrinsically 1D (i1D) structures in images.
Sofar, we only considered the solution of the Laplace equation for $s = 0$. The extension to $s > 0$ results in a set of operators, called Poisson kernels and conjugated Poisson kernels, respectively, which not only form a Riesz triple with quadrature phase relation as the components in equation (19), but transform the monogenic signal into a representation which is called scale-space representation. In fact, the s-component is a scale-space coordinate. A scale-space is spanned by $(\mathbf{x}, s)$. This is a further important result because for nearly 50 years only the Gaussian scale-space, which results as a solution of the heat equation, has been considered as scale-space for image processing. One advantage of the Poisson scale-space in comparison to the Gaussian scale-space is its alibity to naturally embed the complete local spectral analysis into a scale-space theory. Regrettably, this nice theory derived from Clifford analysis framework is relevant only for i1D signals, yet. The extension to i2D structures in images like curved lines/edges, crosses or any other more general structure will be matter of future research. Nevertheless, in [9] a so-called structure multivector is proposed from which a rich set of features can be derived for a special model of an i2D structure.

3 Knowledge Based Neural Computing

Neural nets are computational tools for learning certain competences of a robot vision system within the perception-action cycle. A net of artificial neurons as primitive processing units can learn for instance classification, function approximation, prediction or control. Artificial neural nets are determined in their functionality by the kind of neurons used, their topological arrangement and communication, and the weights of the connections. By embedding neural computing into the framework of GA, any prior algebraic knowledge can be used for increasing the performance of the neural net. The main benefit we get from this approach is constraining the decision space and, thus, preventing the curse of dimensionality. From this can follow faster convergence of learning and increased generalization capability. For instance, if the task is learning the parameters of a transformation group from noisy data, then noise does not follow the algebraic constraints of the transformation group and will not be learned.
In subsection 3.1 we will consider such type of problems under orthogonal constraints. Another advantage of embedding neural computing into geometric algebras is related to learning of functions. Neural nets composed by neurons of perceptron type - this is the type of neurons we will consider here - are able to learn nonlinear functions. If the nonlinear problem at hand is transformed in an algebraic way to a linear one, then learning becomes much easier and with less ressources. We will consider such problems in case of manifold learning in subsection 3.1 and in case of learning decision boundaries in subsection 3.2. For more details see [17].

3.1 The Spinor Clifford Neuron

In this subsection we will consider the embedding of perceptron neurons into a chosen geometric algebra. The output, y, of such neuron for a given input vector $\mathbf{x}$ and a weight vector $\mathbf{w}$, $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$, is given by

$$y = g\left(f(\mathbf{x}; \mathbf{w})\right). \tag{22}$$

The nonlinear function $g : \mathbb{R} \longrightarrow \mathbb{R}$ is called activation function. It shall be omitted in the moment, say, by setting it to the identity. More important is the propagation function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$,

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + \theta, \tag{23}$$

where $\theta \in \mathbb{R}$ is a bias (threshold). Obviously, one single neuron can learn a linear function, represented by equation (23), as linear regression task by minimizing the sum of squared errors over all samples $(\mathbf{x}^j, r^j)$ from a sufficiently composed sample set (universe). We assume a supervised training scheme where $r^j \in \mathbb{R}$ is the requested output of the neuron. By arranging a certain number of neurons in a single layer fully connected to the input vector x, we will get a single layer perceptron network (SLP). A SLP obviously enables task sharing in a parallel fashion and, hence, can perform a multi-linear regression. If $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, then

$$\mathbf{y} = W\mathbf{x} \tag{24}$$

is representing a linear transformation by the weight matrix W.
After introducing the computational principles of a real valued neuron, we will extend now our neuron model by embedding it into the GA $\mathbb{R}_{p,q}, p + q = n$, see [5]. Hence, for $\mathbf{x}, \mathbf{w}, \theta \in \mathbb{R}_{p,q}$ the propagation function $\mathbf{f} : \mathbb{R}_{p,q} \longrightarrow \mathbb{R}_{p,q}$ reads

$$\mathbf{f}(\mathbf{x}) = \mathbf{w}\mathbf{x} + \theta \tag{25}$$

where $\mathbf{w}\mathbf{x}$ should be the geometric product instead of the scalar product of equation (23). A neuron with a propagation function according to equation (25) is called a Clifford neuron. The superiority of equation (25) over equation (23) follows from explicitly introducing a certain algebraic model as constraint by choosing $\mathbb{R}_{p,q}$ for learning the weight matrix of equation (24) instead of additionally learning the required constraints. It is obviously more advantageous to use a model than to perform its simulation. This leads in addition to a reduction of the required resources (neurons, connections). By explicitly introducing algebraic constraints, statistical learning will become a simpler task.
We will further extend our model. If the weight matrix W in equation (24) is representing an orthogonal transformation, we can introduce the constraint of an orthogonal transformation group. This is done by embedding the propagation function now into $\mathbb{R}^+_{p,q}$. Instead of equation (25) we get for $\mathbf{f}$: $\mathbb{R}^+_{p,q} \longrightarrow \mathbb{R}^+_{p,q}$:

$$\mathbf{f}(\mathbf{x}) = \mathbf{w}\mathbf{x}\tilde{\mathbf{w}} + \theta. \tag{26}$$

We will call a neuron with such propagation function a spinor Clifford neuron because the spinor product of the input vector with one weight vector is computed. The representation of an orthogonal transformation by a spinor [12] has several computational advantages in comparison to a matrix representation which should not be explicitly discussed here. It should only be mentioned that one single spinor Clifford neuron is sufficient to learn any transformation group if the right algebra $R_{p,q}$ is chosen where this transformation group becomes a spin group. To mention one extreme example, in [4], the learning of the cross-ratio by one spinor Clifford neuron in $\mathbb{R}_{1,2}$, i.e. the conformal model of the complex plane, has been reported. The neuron had to learn that Möbius transformation as special conformal transformation which keeps the cross-ratio of four points in the complex plane $\mathbb{R}_{0,1}$ invariant. But in contrast to $\mathbb{R}_{0,1}$, in $\mathbb{R}_{1,2}$ the Möbius transformation is an orthogonal one.

3.2 The Hypersphere Neuron

In subsection 3.1 we neglected the activation function in describing our algebraic neuron models. This is possible if a kind of function learning is the task at hand. In case of a classification task the non-linear activation function is mandatory to complete the perceptron to a non-linear computing unit which in the trained state represents a linear decision boundary in $\mathbb{R}^n$. The decision boundary enables solving a 2-class decision problem by dividing the sample space in two half-spaces. Regrettably, the higher the dimension n of the feature vectors, the less likely a 2-class problem can be treated with a linear hyperplane of $\mathbb{R}^n$. Therefore, parallel (SLP) or sequential (MLP - multilayer perceptron) compositions of neurons are necessary for modelling non-linear decision boundaries.
One special case is a hypersphere decision boundary. This can be only very inefficiently modelled by using perceptrons. But spheres are the geometric basis entities of conformal geometry. That is, in conformal geometry we can operate on hyperspheres in a linear manner just as operating on points in Euclidean geometry (points are the basis entities of Euclidean geometry).
In the last few years the conformal geometric algebra (GA) $\mathbb{R}_{p+1,q+1}$ [13] became an important embedding framework of different problems relevant to robot vision because of several attractive properties. We will come back to that point in section 4. As mentioned above, the conformal group $C(p,q)$, which is highly non-linear in $\mathbb{R}^{p,q}$ transforms to a representation which is isomorphic to the orthogonal group $O(p+1,q+1)$ in $\mathbb{R}_{p+1,q+1}$. Second, a subspace of the conformal space $\mathbb{R}^{n+1,1}$ which is called horosphere, N_e^n, is a non-Euclidean model of the Euclidean space $\mathbb{R}^n$ with the remarkable property of metrical equivalence. This is the property we want to exploit for constructing the hypersphere neuron. Here we need in fact only the vector space $\mathbb{R}^{n+1,1} = \langle\mathbb{R}_{n+1,1}\rangle_1$ of the CGA.
Any point $\mathbf{x} \in \mathbb{R}^n$ transforms to a point $\underline{\mathbf{x}} \in \mathbb{R}^{n+1,1}$ with $\underline{\mathbf{x}}^2 = 0$. That is, points are represented as null vectors on the horosphere. Because hyperspheres are the basis entities of $\mathbb{R}^{n+1,1}$, a point $\underline{\mathbf{x}}$ can be seen as a degenerate hypersphere $\underline{\mathbf{s}}$ with radius zero.
Let be $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ two points with Euclidean distance $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$ and let be

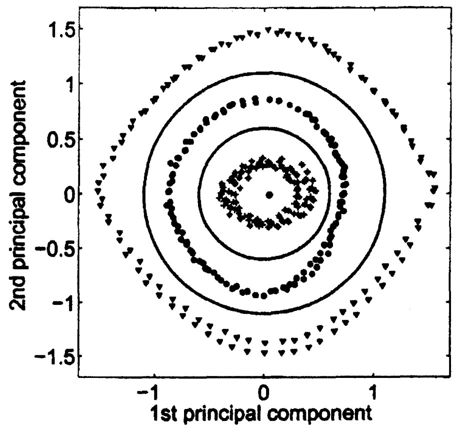

Figure 3: The decision boundaries and the first two principal components of three toy objects rotated by 360^o in steps of one degree.

$\underline{x}, \underline{y} \in N_e^n \subset \mathbb{R}^{n+1,1}$ with distance $d(\underline{x}, \underline{y}) = \underline{x} \cdot \underline{y}$ (scalar product), then

$$d(\underline{x}, \underline{y}) = -\frac{1}{2} d^2(x, y). \tag{27}$$

Therefore, the distance of a point $\underline{x}$ to hypersphere $\underline{s}$ will be

$$d(\underline{x}, \underline{s}) : \begin{cases} > 0 & \text{if} \quad \underline{x} \quad \text{is outside } \underline{s} \\ = 0 & \text{if} \quad \underline{x} \quad \text{is on } \underline{s} \\ < 0 & \text{if} \quad \underline{x} \quad \text{is inside } \underline{s}. \end{cases} \tag{28}$$

That distance measure is used for designing the propagation function of a hypersphere neuron [1]. If the parameters of the hypersphere are interpreted as the weights of a perceptron, then by embedding any data points into $\mathbb{R}^{n+1,1}$, the decision boundary will be a hypersphere. In fact, the hypersphere neuron subsumes the classical perceptron because a hypersphere with infinite radius is a hyperplane.
To simplify the implementation we take advantage of the equivalence of the scalar products in $\mathbb{R}^{n+1,1}$ and in $\mathbb{R}^{n+2}$. This enables a data coding which makes the hypersphere neuron to a perceptron with a second bias component. For an example see figure (3).

4 Pose Estimation of Free-form Objects

As a rather complex example for using GA in robot vision we will report on estimating the pose of an object which is moving in 3D space. In that example meet computer vision and robotics very obviously. As pose we denote position and orientation. Hence, pose estimation means estimating the parameters of the special Euclidean transformation in

space by observations of the rigid body motion in the image plane. Pose estimation is a basic task of robot vision which can be used in several respects as part of more complex tasks as visual tracking, homing or navigation. Although there is plenty of solutions over the years, only CGA [13] gives a framework which is adequate to the problem at hand [15].
The diversity of approaches known sofar results from the hidden complexity of the problem. Estimation of the parameters of the special Euclidean transformation, which is composed by rotation and translation, has to be done in a projective scenario. But the coupling of projective and special Euclidean transformations, both with non-linear representations in the Euclidean space, results in a loss of the group properties. Another point is how to model the object which serves as reference. In most cases a set of point features is used. But this does not enable to exploit internal constraints contained in higher order entities as lines, planes, circles, etc. Finally, it can be distinguished between 2D and 3D representations of either model data and measurement data.
To be more specific, the task of pose estimation is the following: Given a set of geometric features represented in an object centered frame and given the projections of these features onto the image plane of a camera, then, for a certain parametrized projection model, determine the parameters of the rigid body motion between two instances of an object centered frame by observations in a camera centered frame.
In our scenario we are assuming a 2D-3D approach, i.e. having 2D measurement data from a calibrated full perspective monocular camera and 3D model data. As model we are taking either a set of geometric primitives as points, lines, planes, circles or spheres, or more complex descriptions as a kinematic chain (coupled set of piecewise rigid parts) or free-form curves/surfaces given as CAGD-model. From the image features we are projectively reconstructing lines or planes in space. Now the task is to move the reference model in such a way that the spatial distance of the considered object features to the projection rays/planes becomes zero. This is an optimization task which is done by a gradient descent method on the spatial distance as error measure. Hence, we prevent minimizing any error measure directly on the manifold.

4.1 Pose Estimation in Conformal Geometric Algebra

We cannot introduce here the conformal geometric algebra [13] because of limited space. There is plenty of good introductions, see e.g. the PhD thesis [15] with respect to an extended description of pose estimation in that framework.
We are using the CGA $\mathbb{R}_{4,1}$ of the Euclidean space $\mathbb{R}^3$. If $\mathbb{R}^3$ is spanned by the unit vectors $\mathbf{e}_i$, $i = 1, 2, 3$, then $\mathbb{R}_{4,1}$ is the algebra of the augmented space $\mathbb{R}^{4,1} = \mathbb{R}^{3,0} \oplus \mathbb{R}^{1,1}$ with the additional basis vectors $\mathbf{e}_o$ (representing the point at origin) and $\mathbf{e}$ (representing the point at infinity). The basis $\{\mathbf{e}_o, \mathbf{e}\}$ constitutes a null basis. From this follows that the representation of geometric entities by null vectors is the characteristic feature of that special conformal model of the Euclidean space.
The advantages we can profit from within our problem are the following. First, $\mathbb{R}_{4,1}$

constitutes a unique framework for affine, projective and Euclidean geometry. If $\mathbb{R}_{3,0}$ is the geometric algebra of the 3D Euclidean space and $\mathbb{R}_{3,1}$ is that one of the 3D projective space, then we have the following relations between these linear spaces:

$$\mathbb{R}_{3,0} \subset \mathbb{R}_{3,1} \subset \mathbb{R}_{4,1}.$$

Because the special Euclidean transformation is a special affine transformation, we can handle either kinematic, projective or metric aspects of our problem in the same algebraic frame. Higher efficiency is reached by simply transforming the representations of the geometric entities to the respective partial space.
Second, the special Euclidean group $SE(4,1)$ is a subgroup of the conformal group $C(4,1)$, which is an orthogonal group in $\mathbb{R}^{+}_{4,1}$. Hence, the members of $SE(4,1)$, which we call motors, $\mathbf{M}$, are spinors representing a general rotation. Any entity $\underline{\mathbf{u}} \in \mathbb{R}_{4,1}$ will be transformed by the spinor product,

$$\underline{\mathbf{u}}' = \mathbf{M}\underline{\mathbf{u}}\widetilde{\mathbf{M}}. \tag{29}$$

Let be $\mathbf{R}$ a rotor as representation of pure rotation and let be $\mathbf{T}$ another rotor, called translator, representing translation in $\mathbb{R}^3$, then any $g \in SE(4,1)$ is given by the motor

$$\mathbf{M} = \mathbf{T}\mathbf{R} \tag{30}$$

with $\mathbf{R}, \mathbf{T}, \mathbf{M} \in \mathbb{R}^{+}_{4,1}$. Motors have some nice properties. They concatenate multiplicatively. If e.g. $\mathbf{M} = \mathbf{M}_2\mathbf{M}_1$, then

$$\underline{\mathbf{u}}'' = \mathbf{M}\underline{\mathbf{u}}\widetilde{\mathbf{M}} = \mathbf{M}_2\underline{\mathbf{u}}'\widetilde{\mathbf{M}}_2 = \mathbf{M}_2\mathbf{M}_1\underline{\mathbf{u}}\widetilde{\mathbf{M}}_1\widetilde{\mathbf{M}}_2. \tag{31}$$

Furthermore, motors are linear operations (as used also in context of spinor Clifford neurons). That is, we can exploit the outermorphism [11], which is the preservation of the outer product under linear transformation.
This leads directly to the third advantage of CGA. The incidence algebra of projective geometry generalizes in CGA. If $\underline{\mathbf{s}}_1, \underline{\mathbf{s}}_2 \in \langle\mathbb{R}_{4,1}\rangle_1$ are two spheres, then their outer product is a circle, $\underline{\mathbf{z}} \in \langle\mathbb{R}_{4,1}\rangle_2$,

$$\underline{\mathbf{z}} = \underline{\mathbf{s}}_1 \wedge \underline{\mathbf{s}}_2. \tag{32}$$

If the circle is undergoing a rigid motion, then

$$\underline{\mathbf{z}}' = \mathbf{M}\underline{\mathbf{z}}\widetilde{\mathbf{M}} = \mathbf{M}(\underline{\mathbf{s}}_1 \wedge \underline{\mathbf{s}}_2)\widetilde{\mathbf{M}} = \mathbf{M}\underline{\mathbf{s}}_1\widetilde{\mathbf{M}} \wedge \mathbf{M}\underline{\mathbf{s}}_2\widetilde{\mathbf{M}} = \underline{\mathbf{s}}_1' \wedge \underline{\mathbf{s}}_2'. \tag{33}$$

In very contrast to the (homogeneously extended) Euclidean space, we can handle the rigid body motion not only as linear transformation of points but of any entities derived from points or spheres. These entities are no longer only set conceptions as subspaces in a vector space but algebraic entities. This enables also to handle any free-form object as one algebraic entity which can be used for gradient based pose estimation in the same manner as points. That is a real cognitive approach to the pose problem.

Finally, we can consider the special Euclidean group $SE(4,1)$ as Lie group and are able to estimate the parameters of the generating operator of the group member. This approach leads to linearization and very fast iterative estimation of the parameters. Although this approach is also applicable to points in an Euclidean framework, in CGA it is applicable to any geometric entity built from points or spheres.
The generating operator of a motor is called twist, $\Psi \in se(4,1)$. The model of the motor as a Lie group member changes in comparison to equation (30) to

$$\mathbf{M} = \mathbf{T}\mathbf{R}\widetilde{\mathbf{T}}. \tag{34}$$

This equation, which expresses a general rotation as Lie group member in $\mathbb{R}_{4,1}$, can be easily interpreted. The general rotation is performed as normal rotation, $\mathbf{R}$, in the rotation plane $\mathbf{l} \in \langle\mathbb{R}_3\rangle_2$, which passes the origin, by an angle $\theta \in \mathbb{R}$, after correcting the displacement $\mathbf{t} \in \langle\mathbb{R}_3\rangle_1$ of the rotation axis $\mathbf{l}^* \in \langle\mathbb{R}_3\rangle_1$ from the origin with help of the translator $\widetilde{\mathbf{T}}$. Finally, the displacement has to be reconstructed by the translator $\mathbf{T}$. From this follows equation (34) in terms of the parameters of the rigid body motion in $\mathbb{R}_3$,

$$\mathbf{M} = \exp\left(\frac{\mathbf{et}}{2}\right) \exp\left(-\frac{\theta}{2}\mathbf{l}\right) \exp\left(-\frac{\mathbf{et}}{2}\right). \tag{35}$$

This factorized representation can be compactly written as

$$\mathbf{M} = \exp\left(-\frac{\theta}{2}\Psi\right). \tag{36}$$

Here

$$\Psi = \mathbf{l} + \mathbf{e}(\mathbf{t}\cdot\mathbf{l}) \tag{37}$$

is the twist of rigid body motion. Geometrically interpreted is the twist representing the line $\underline{\mathbf{l}} \in \langle\mathbb{R}_{4,1}\rangle_2$ around which the rotation is performed in $\mathbb{R}_{4,1}$.
Now we return to our problem of pose estimation. The above mentioned optimization problem is nothing else as a problem of minimizing the spatial distance of e.g. a projection ray to a feature on the silhouette of the reference model. If this distance is zero, then a certain geometric constraint is fulfilled. These constraints are either collinarity or coplanarity for points, lines and planes, or tangentiality in case of circles or spheres.
For instance, the collinearity of a point $\underline{\mathbf{x}} \in \langle\mathbb{R}_{4,1}\rangle_1$, after beeing transformed by the motor $\mathbf{M}$, with a projection line $\underline{\mathbf{l}}_x$ is written as their vanishing commutator product,

$$(\mathbf{M}\underline{\mathbf{x}}\widetilde{\mathbf{M}}) \underline{\times} \underline{\mathbf{l}}_x = 0. \tag{38}$$

This equation has to be written in more details to see its coupling with the observation of a point x in the image plane of the camera:

$$\lambda\left((\mathbf{M}\underline{\mathbf{x}}\widetilde{\mathbf{M}}) \underline{\times} (\mathbf{e} \wedge (\mathbf{O} \wedge x))\right) \cdot \mathbf{e}_+ = 0. \tag{39}$$

This in fact is the complete pose estimation problem written as a symbolic dense but nevertheless algebraic correct equation. The outer product $\mathbf{O} \wedge x$ of the image point x with the optical center $\mathbf{O}$ of the camera results in the projection ray representation in projective space $\mathbb{R}_{3,1}$. By wedging it with $\mathbf{e}$, we transform the projection ray to the conformal space $\mathbb{R}_{4,1}$, hence, $\underline{\mathbf{l}}_x = \mathbf{e} \wedge \mathbf{O} \wedge x$. Now the collinearity constraint can be computed in $\mathbb{R}_{4,1}$. Finally, to assign the constraint an Euclidean distance zero, the expression has to be transformed to the Euclidean space $\langle \mathbb{R}_3 \rangle_1$ by computing the inner product with the basis vector $\mathbf{e}_+$ which is derived from the null basis and by scaling with $\lambda \in \mathbb{R}$. In reality the commutator product will take on a minimum as a result of the optimization process.

4.2 Twist Representations of Free-form Objects

Sofar we only considered the pose estimation problem on the base of modelling a rigid object by a set of different order geometric entities. In robotics exists another important object model which is piecewise rigid. A kinematic chain is the formalization of several rigid bodies coupled by either revolute or prismatic joints. If pose estimation is applied to observatons of e.g. a robot arm for controlling grasping movements, this is the adequate model. Each of the parts of an arm is performing movements in mutual dependence. Hence, the motion of the j-th joint causes also motions of the preceding ones. If $\underline{\mathbf{u}}_j \in \mathbb{R}_{4,1}$ is a geometric entity attached to the j-th segment, e.g. a fingertip, its net displacement caused by a motor $\mathbf{M}$, represented in the (fixed) base coordinate system, is given by

$$\underline{\mathbf{u}}'_j = \mathbf{M}(\mathbf{M}_1 ... \mathbf{M}_j \underline{\mathbf{u}}_j \widetilde{\mathbf{M}}_j ... \widetilde{\mathbf{M}}_1)\widetilde{\mathbf{M}}. \tag{40}$$

Here, the motors $\mathbf{M}_i$ are representing the constrained motion of the i-th joint.
While in the homogeneous space $\mathbb{R}^4$ this equation is limited to points as moving entities and in the framework of the motor algebra [2] lines and planes can be considered in addition, there is no such restriction in CGA.
The idea of the kinematic chain can be further generalized to a generator of free-form shape. We will consider the trajectory caused by the motion of the entity $\underline{\mathbf{u}}$ in space as the orbit of a multi-parameter Lie group of $SE(4,1)$. This multi-parameter Lie group represents a possibly very complex general rotation in $\mathbb{R}_{4,1}$ which is generated by a set of nested motors, contributing each with its own constrained elementary general rotation. If the entity $\underline{\mathbf{u}}$ would be a point, then the orbit can be a space curve, a surface or a volume. But $\underline{\mathbf{u}}$ can also be of higher order (e.g. a tea pot). Hence, the generated orbit may be of any complexity.
This kinematic model of shape in CGA [16] has not been known before, although there is a long history of relating kinematics and geometry, e.g. with respect to algebraic curves, or of the geometric interpretation of Lie groups.
The mentioned generalization of the model of a kinematic chain is twofold. First, the rotation axes of the motors must not be positioned at the periphery of the shape but may be (virtually) located anywhere. Second, several different oriented axes may be fixed at

the same location.
The principle of constructing higher order 3D curves or surfaces as orbit of a point which is controlled by only a few coupled twists as generators is simple. Let be $\underline{\mathbf{x}}_z$ an arbitrary point on a circle $\underline{\mathbf{z}}$, that is $\underline{\mathbf{x}}_z \cdot \underline{\mathbf{z}} = 0$, and let be $\mathbf{M}(\underline{\mathbf{l}};\theta)$ the corresponding motor, then for all $\theta \in [0,...,2\pi]$

$$\underline{\mathbf{x}}_z^{(1)} = \mathbf{M}\underline{\mathbf{x}}_z\widetilde{\mathbf{M}} \tag{41}$$

generates the circle $\underline{\mathbf{z}}$. If the motor is representing a pure translator, then any arbitrary oriented line $\underline{\mathbf{l}}$ will be generated. Such primitive curves will be called 3D-1twist curves. By coupling a second twist to the first one, either a 3D-2twist curve or surface will be generated. In fact, already this simple system has a lot of degrees of freedom which enable generating quite different figures [15]. This can be seen by the following equation:

$$\underline{\mathbf{x}}_c^{(2)} = \mathbf{M}_2\mathbf{M}_1\underline{\mathbf{x}}_c\widetilde{\mathbf{M}}_1\widetilde{\mathbf{M}}_2 \tag{42}$$

with

$$\mathbf{M}_i(\underline{\mathbf{l}}_i;\lambda_i,\theta_i) = \exp\left(-\frac{\lambda_i\theta_i}{2}\Psi_i\right), \tag{43}$$

Here $\lambda_i \in \mathbb{R}$ is the ratio of the angular frequency, $\theta_i \in [\alpha_1,...,\alpha_2]$ is the angular segment which is covered by θ_i and finally Ψ_i defines the position and orientation of the general rotation axis $\underline{\mathbf{l}}_i$ and the type of motion, that is rotation-like, translation-like, or a mixed form.
Finally, the twist model of a closed shape is equivalent to the well-known Fourier representation, we started with this survey [17]. Since the Fourier series expansion of a closed 3D curve $C(\phi)$ can be interpreted as the action of an infinite set of rotors, $\mathbf{R}_k^{\phi}$, fixed at the origin of $\langle\mathbb{R}_3\rangle_1$, onto phase vectors $\mathbf{p}_k$, there is no need of using CGA. A plane closed curve is given by

$$C(\phi) = \lim_{N\to\infty}\sum_{k=-N}^{N}\mathbf{p}_k \exp\left(\frac{2\pi k\phi}{T}\mathbf{l}\right) = \lim_{N\to\infty}\sum_{k=-N}^{N}\mathbf{R}_k^{\phi}\mathbf{p}_k\widetilde{\mathbf{R}}_k^{\phi} \tag{44}$$

The phase vectors $\mathbf{p}_k$ are the Fourier series coefficients. Instead of the imaginary unit of the harmonics, the unit bivector $\mathbf{l} \in \mathbb{R}_3, \mathbf{l}^2 = -1$, is used which defines the rotation plane of the phase vectors. Based on the assumption that any closed 3D curve can be reconstructed from three orthogonal projections, a spatial curve is represented by

$$C(\phi) = \lim_{N\to\infty}\sum_{m=1}^{3}\sum_{k=-N}^{N}\mathbf{p}_k^m \exp\left(\frac{2\pi k\phi}{T}\mathbf{l}_m\right). \tag{45}$$

This scheme can be extended to free-form surfaces as well and has been used for pose estimation in the presented framework. In that respect the inverse Fourier transform of discretized curves/surfaces was applied. The necessary transformation of the Euclidean

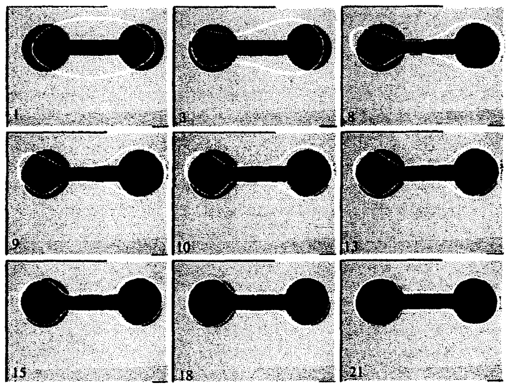

Figure 4: Regularization of iterative pose estimation by Fourier representation (numbers indicate the iteration steps).

Fourier representation, $C_E(\phi)$, to a conformal one, $C_C(\phi)$, can simply be done by the following operation:

$$C_C(\phi) = \mathbf{e} \wedge (C_E(\phi) + \mathbf{e}_-) \tag{46}$$

where $\mathbf{e}_-$ stands for the homogeneous coordinate of the projective space $\mathbb{R}_{3,1}$ and wedging with $\mathbf{e}$ again realizes the transformation from $\mathbb{R}_{3,1}$ to $\mathbb{R}_{4,1}$.
One advantage of using the Fourier interpretation of the twist approach in pose estimation is the possibility of regularizing the optimization in case of non-convex objects. This is demonstrated in figure (4). During the iteration process, which needs for that object only a few milliseconds, successively more Fourier coefficients are used. Hence, getting stuck in local minima is prevented.
The presented methods used in pose estimation clearly demonstrate the strength of the applied algebraic model of conformal geometric algebra.

5 Conclusions

We have demonstrated the application of geometric algebra as universal mathematical language for modelling in robot vision. Here we will summarize some general conclusions. First, GA supports generalizations of representations. In section 4 we used the stratification of spaces of different geometry. This enables simple switching between different aspects of a geometric entity. Instead of points, in certain GA as CGA, higher order entities take on the role of basis entities from which object concepts with algebraic properties can be constructed. We have shown that kinematics, shape theory and Fourier theory are unified in the framework of CGA. Besides, in section 2 we could handle multi-dimensional functions in a linear signal theory.
Second, the transformation of non-linear problems to linear ones is another important advantage. This could be demonstrated in all presented application fields. In learning

theory this enables designing algebraically constrained knowledge based neural nets.
Third, GA is a mathematical language which supports symbolic dense formulations with algebraic meaning. We are able to lift up representations where besides the above mentioned qualitative aspects also reduced computational complexity results. This is important in real-time critical applications as robot vision.
We can state that the progress we made in robot vision could only be possible on the base of using geometric algebra and this enforces its use also in other application fields.

6 Acknowledgements

We acknowledge the European Community, Deutsche Forschungsgemeinschaft and Studienstiftung des deutschen Volkes for supporting the research which is reported in this paper. Especially contributions of Thomas Bülow, Michael Felsberg, Sven Buchholz, Vladimir Banarer, Bodo Rosenhahn and Christian Perwass contributed to the reported results. Our thanks are also to Prof. Hitzer, Prof. Nagaoka and Dr. Ishi for organizing the stimulating conference in Kyoto.

References

[1] V. Banarer, C. Perwass, and G. Sommer. The hypersphere neuron. In *Proc. 11th European Symposium on Artificial Neural Networks, ESANN 2003, Bruges*, pages 469–474. d-side publications, Evere, Belgium, 2003.

[2] E. Bayro-Corrochano, K. Daniilidis, and G. Sommer. Motor algebra for 3d kinematics: The case of hand-eye calibration. *Journal of Mathematical Imaging and Vision*, 13:79–100, 2000.

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

[4] S. Buchholz and G. Sommer. Learning geometric transformations with Clifford neurons. In G. Sommer and Y. Zeevi, editors, *2nd International Workshop on Algebraic Frames for the Perception-Action Cycle, AFPAC 2000, Kiel*, volume 1888 of *LNCS*, pages 144–153. Springer-Verlag, 2000.

[5] S. Buchholz and G. Sommer. Introduction to neural computation in Clifford algebra. In G. Sommer, editor, *Geometric Computing with Clifford Algebras*, pages 291–314. Springer-Verlag, Heidelberg, 2001.

[6] T. Bülow. Hypercomplex spectral signal representations for the processing and analysis of images. Technical Report Number 9903, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, August 1999.

[7] T. Bülow and G. Sommer. Hypercomplex signals - a novel extension of the analytic signal to the multidimensional case. *IEEE Transactions on Signal Processing*, 49(11):2844–2852, 2001.

[8] M. Felsberg. Low-level image processing with the structure multivector. Technical Report Number 0203, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, März 2002.

[9] M. Felsberg and G. Sommer. The monogenic signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, 2001.

[10] R.V.L. Hartley. A more symmetrical Fourier analysis applied to transmission problems. *Proc. IRE*, 30:144–150, 1942.

[11] D. Hestenes. The design of linear algebra and geometry. *Acta Appl. Math.*, 23:65–93, 1991.

[12] D. Hestenes, H. Li, and A. Rockwood. New algebraic tools for classical geometry. In G. Sommer, editor, *Geometric Computing with Clifford Algebras*, pages 3–23. Springer-Verlag, Heidelberg, 2001.

[13] H. Li, D. Hestenes, and A. Rockwood. Generalized homogeneous coordinates for computational geometry. In G. Sommer, editor, *Geometric Computing with Clifford Algebras*, pages 27–59. Springer-Verlag, Heidelberg, 2001.

[14] C. Perwass and D. Hildenbrand. Aspects of geometric algebra in euclidean, projective and conformal space. Technical Report Number 0310, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, September 2003.

[15] B. Rosenhahn. Pose estimation revisited. Technical Report Number 0308, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, September 2003.

[16] B. Rosenhahn, C. Perwass, and G. Sommer. Free-form pose estimation by using twist representations. *Algorithmica*, 38:91–113, 2004.

[17] G. Sommer. The geometric algebra approach to robot vision. Technical Report Number 0304, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, September 2003.