Postprint of: Aerts D., Czachor M., De Moor B., Geometric analogue of holographic reduced representation, Journal of Mathematical Psychology, Vol. 53, Iss. 5 (2009), pp. 389-398, DOI: 10.1016/j.jmp.2009.02.005

Geometric Analogue of Holographic Reduced Representation

Diederik Aerts [1], Marek Czachor [2,3], and Bart De Moor [3]
[1] *Centrum Leo Apostel (CLEA) and Foundations of the Exact Sciences (FUND)*
Brussels Free University, 1050 Brussels, Belgium
[2] *Katedra Fizyki Teoretycznej i Informatyki Kwantowej*
Politechnika Gdańska, 80-952 Gdańsk, Poland
[3] *ESAT-SCD, Katholieke Universiteit Leuven,*
3001 Leuven, Belgium

Holographic reduced representations (HRR) are based on superpositions of convolution-bound n-tuples, but the n-tuples cannot be regarded as vectors since the formalism is basis dependent. This is why HRR cannot be associated with geometric structures. Replacing convolutions by geometric products one arrives at reduced representations analogous to HRR but interpretable in terms of geometry. Variable bindings occurring in both HRR and its geometric analogue mathematically correspond to two different representations of $Z_2 \times \cdots \times Z_2$ (the additive group of binary n-tuples with addition modulo 2). As opposed to standard HRR, variable binding performed by means of geometric product allows for computing exact inverses of all nonzero vectors, a procedure even simpler than approximate inverses employed in HRR. The formal structure of the new reduced representation is analogous to cartoon computation, a geometric analogue of quantum computation.

I. INTRODUCTION

Reduced representations of cognitive structures are based essentially on two operations (binding and superposing) whose algebraic realizations vary from model to model. In [1], where matrices representing roles act on vectors representing fillers, binding corresponds to matrix multiplication and superposition to vector addition. In tensor representations [2] roles and fillers are represented by vectors which are bound by means of tensor products. The resulting simple tensors are superposed by addition. In holography-inspired memory models [3, 4, 5, 6, 7, 8, 9, 10] binding is represented by convolution. Replacing tensor products by circular convolutions one arrives at holographic reduced representations (HRR) [11, 12]. Restricting frequency-domain HRR to a subspace and switching to appropriately defined 'logarithmic' variables one obtains binary spatter codes (BSC) [13, 14, 15], with binary strings of length n bound by n-dimensional sums mod 2 and superpositions modeled by majority-rule sums. Finally, in quantum computation (QC) [16] bits are bound into n-bit numbers by means of tensor products of two-dimensional complex vectors called qubits. QC is mathematically similar to tensor-product reduced representations, but differences occur at interpretational levels [17].

Convolutions may be regarded as basis-dependent lossy compressions of the tensor product. The degree of compression can be estimated by means of dimensional analysis. In particular, circular convolutions occurring in HRR map pairs of n-tuples into n-tuples. In contrast, the ordinary convolution of two n-tuples is a $(2n-1)$-tuple, and an analogous tensor product would produce a n^2-tuple. These facts explain efficiency and usefulness of HRR in applications [12].

In spite of what one can often read in the literature, a convolution of two *vectors* is not well defined. This means that having two vectors, that is — geometric objects, we cannot unambiguously identify a geometric object corresponding to their circular convolution. This seems to be a drawback, at least at the conceptual level. Geometry of some sort is at the roots of visualization, and visualization seems important for mathematical *understanding* and *proving* [18, 19, 20, 21]. Hence the question: Is it possible to replace circular convolution by something similar but geometrically meaningful? If so, is there a relation to HRR?

We will argue that the most natural choice is to replace tensor products by *geometric products* [22], and not by circular convolutions. Geometric products, similarly to circular convolutions, preserve dimensionality at the level of *multivectors*. Multivectors are superpositions of *blades*, geometric-product analogues od simple tensors. Geometric products are also 'exponents', in a sense that will be made precise later, of n-dimensional sums mod 2. In consequence, geometric-product binding is in a unique relation to the binding employed in BSC, and the latter is a form of HRR.

Coding based directly on geometric products was recently applied to QC [23, 24, 25] (a somewhat less direct way of linking geometric products with QC was used earlier in [26]). As it turned out, all quantum algorithms of the standard formalism have geometric analogues. However, as opposed to standard QC that requires quantum mechanical implementations, the formalism based on geometric algebra (GA) requires *geometry* and not quantum mechanics. In principle, any system involving some geometry (Euclidean or not) is a candidate for implementation of a quantum algorithm. Systems where HRR are applicable might therefore, at least in principle, perform quantum algorithms.

The concept of GA is not new — it appeared in the 19th century works of Grassmann [27] and Clifford [28] — but geometric insights behind GA were forgotten for almost a century. At the end of 1960s the subject was revived with the works of Hestenes [29]. Today the Hestenes system [30, 31, 32] has found applications (cf. [33, 34, 35, 36, 37, 38, 39, 40]) to topics as diverse as black holes, cosmology, quantum mechanics, quantum field theory, supersymmetry, beam dynamics, computer vision, robotics, protein folding, neural networks, computer aided design, and recently — quantum computation. The link between HRR and GA suggests that the next step is to reformulate in a GA way the cognitive science.

The paper is organized as follows. Section II introduces GA and its basic constructions (multivectors, invertibility, coding based on blades) and explains why GA naturally formalizes relations between geometric objects. In Section III we compare geometric product with circular convolution and explain why the latter cannot be interpreted in geometric terms. In Section IV we discuss the Fourier-space convolution algebra and show that it is in a one-to-one relation with the algebra of commuting matrices. In Section V we explain why variable binding in Fourier-space HRR is a representation, in group-theoretic sense, of the group $Z_2 \times \cdots \times Z_2$. Then, in Section VI we show that blades form a projective representation the same group and thus convolution binding is naturally represented in GA. We illustrate the new reduced representation in Section VII by reformulating the example Kanerva gave as an illustration of his BSC. Finally, in Section IX we reformulate the same example by means of a matrix representation of GA.

II. ALGEBRAIC REPRESENTATION OF GEOMETRIC RELATIONS

Consider the following set of relations $\approx$ involving two-dimensional basic shapes:

$$-- \;\approx\; ||\approx \square\square \approx 1 \tag{1}$$
$$-| \;\approx\; |-\approx \square \tag{2}$$
$$-\square \;\approx\; \square- \approx | \tag{3}$$
$$|\square \;\approx\; \square| \approx - \tag{4}$$

One can think of them in at least two categories. One is simply a category of *understanding* relations between one- and two-dimensional objects: Square is formed from two orthogonal segments, a segment and a square imply which segment is missing, 1 means identity... Another way of looking at these relations, more in the spirit of Grassmann and Clifford, would be in terms of an *algebra* of geometric objects if associativity is assumed and 1 is treated as a neutral element. Associativity then implies, for example, $-|-\approx\square-\approx-\square\approx|$. Let us make here a side remark that the 'algebra' formalizes a procedure that resembles an IQ test.

The next step is to ask for higher-dimensional generalization and inclusion of *orientation*: Oriented line segments are vectors, plane segments have 'sides', the relations between them will include a positive or negative sign, and we can also add cubes (having 'insides' and 'outsides'), their walls, and even higher dimensional structures.

Orientation makes the algebra noncommutative

$$\rightarrow\uparrow \;\approx\; +\square \tag{5}$$
$$\uparrow\rightarrow \;\approx\; -\square \tag{6}$$
$$\rightarrow\rightarrow \;\approx\; \uparrow\uparrow\approx 1 \tag{7}$$

if we assume that 'first up then right' generates the righthanded orientation, opposite to the lefthanded 'first right then up'. Adding both relations we find $\rightarrow\uparrow+\uparrow\rightarrow\approx 0$. Taking three arrows and using associativity we obtain another, similar rule: $\square\rightarrow+\rightarrow\square\approx 0$. The proof goes as follows

$$\square\rightarrow\approx\rightarrow\uparrow\rightarrow\approx\rightarrow(-\square)\;\approx -\rightarrow\square\;\approx\leftarrow\square \tag{8}$$

However, we cannot simultaneously assume $\rightarrow\rightarrow\approx\uparrow\uparrow\approx 1$ and $\square\square\approx 1$. Indeed, anticommutativity of $\rightarrow\uparrow\approx-\uparrow\rightarrow$ implies

$$\square\square\approx\rightarrow\uparrow\rightarrow\uparrow\approx-\rightarrow\rightarrow\uparrow\uparrow\approx -11\approx -1 \tag{9}$$

so we have to decide at which level to define the algebra. One can also think of this example as a hierarchy of geometric structures with increasing dimensions and different metric signatures. The first level is the pair $\{\rightarrow,\uparrow\}$ and the Euclidean-space signature is $(+,+)$. The second level is $\{\mathbf{1},\rightarrow,\uparrow,\square\}$ and the signature is $(+,+,+,-)$, known from space-time pseudo-Euclidean geometry.

The type of construction we have just outlined led Grassmann and Clifford to the algebra based on the concise formula

$$b_k b_l + b_l b_k = 2\delta_{kl}\mathbf{1}. \tag{10}$$

Here the bs denote orthonormal basis vectors in some n-dimensional real Euclidean space, $\mathbf{1}$ is the neutral element of the algebra, and δ_{kl} is the Kronecker delta. The 2D plane example is reconstructed from (10) if $b_1 = \rightarrow$, $b_2 = \uparrow$, $b_1 b_2 = \square = b_{12}$.

The algebra (10) is known as the Clifford algebra and may be regarded as the grammar of GA. It refers to a concrete basis, but can be reformulated in a basis-free way. Indeed, consider two vectors $x = \sum_{k=1}^{n} x_k b_k$ and $y = \sum_{k=1}^{n} y_k b_k$. Their geometric product reads

$$xy = \underbrace{\sum_{k=1}^{n} x_k y_k \mathbf{1}}_{x\cdot y} + \underbrace{\sum_{k<l} (x_k y_l - y_k x_l) b_k b_l}_{x\wedge y} \tag{11}$$

The geometric product xy is a sum of two terms. The *scalar* $x \cdot y = y \cdot x$ is known as the *inner product*. The *bivector* $x \wedge y = -y \wedge x$ is the *outer product*. In 3D the length of $x \wedge y$ represents the area of the parallelogram spanned by x and y.

In arbitrary dimension the bivector $x \wedge y$ represents an oriented plane segment. Grassmann and Clifford introduced geometric product by means of the basis-independent formula involving a *multivector*

$$xy = x \cdot y + x \wedge y \tag{12}$$

which implies (10) when restricted to the orthonormal basis. Inner and outer product can be defined directly from xy:

$$x \cdot y = \frac{1}{2}(xy + yx), \tag{13}$$

$$x \wedge y = \frac{1}{2}(xy - yx). \tag{14}$$

The most ingenious element of (12) is that it adds two apparently different objects: A scalar and a plane element. This seems 'wrong' but this is precisely what happens when we speak of complex numbers or extend space and time to space-time. Apparently, the person to be blamed for the fact that multivectors may nowadays seem weird is Gibbs [41], who was more famous at the time than Grassmann or Clifford, and spoiled their work by separating the geometric product into two separate operations — losing associativity and invertibility, as we shall see shortly.

Geometric interpretation and visualization of multivectors can be formulated in various ways, and various interpretations can be found in the literature. Perhaps the easiest way of getting used to thinking in GA terms is to browse through the websites devoted to GA (cf. [42]). The approach we found useful for multi-bit problems of QC was a representation in terms of directed colored polylines [25].

Geometric product for vectors x, y, z can be defined by the following rules:

$$(xy)z = x(yz), \tag{15}$$

$$x(y+z) = xy + xz, \tag{16}$$

$$(x+y)z = xz + yz, \tag{17}$$

$$x^2 = |x|^2, \tag{18}$$

where $|x|$ is a positive scalar called the magnitude of x. The rules imply that $x \cdot y$ must be a scalar since

$$xy + yx = |x+y|^2 - |x|^2 - |y|^2. \tag{19}$$

GA allows to speak of inverses of vectors: $x^{-1} = x/|x|^2$. x is invertible (i.e. possesses an inverse) if its magnitude is nonzero. Geometric product of an arbitrary number of invertible vectors is also invertible. The possibility of inverting all nonzero-magnitude vectors is perhaps the most important difference between GA and tensor or convolution algebras.

Geometric products of *different* basis vectors

$$b_{k_1 \ldots k_j} = b_{k_1} \ldots b_{k_j}, \tag{20}$$

$k_1 < \cdots < k_j$, are called blades. In n-dimensional (pseudo-)Euclidean space there are 2^n different blades. This can be seen as follows. Let $\{x_1, \ldots, x_n\}$ be a sequence of bits. Blades in an n-dimensional space can be written as

$$c_{x_1 \ldots x_n} = b_1^{x_1} \ldots b_n^{x_n} \tag{21}$$

where $b_k^0 = \mathbf{1}$, which shows that blades are in a one-to-one relation with n-bit numbers. This observation is at the roots of the GA reformulation of QC introduced in [23]. A general multivector is a linear combination of blades,

$$\psi \;=\; \sum_{x_1 \ldots x_n = 0}^{1} \psi_{x_1 \ldots x_n} c_{x_1 \ldots x_n}, \tag{22}$$

with real coefficients $\psi_{x_1 \ldots x_n}$ [46].

An inverse of a multivector is a well defined notion but not all multivectors are invertible. To find an inverse of a multivector is not an entirely trivial task in general. It resembles an analogous problem of inverting matrices. But all blades and geometric products of invertible vectors are invertible.

III. CIRCULAR CONVOLUTION VS. GEOMETRIC PRODUCT

We have mentioned in Section I that convolutions are not defined on vectors but only on n-tuples. Let us explain this statement in more detail on the example of circular convolution. Circular convolution $x \circledast y$ of the n-tuples $x = (x_0, \ldots, x_{n-1})$, $y = (y_0, \ldots, y_{n-1})$ is defined as

$$(x \circledast y)_j \;=\; \sum_{k=0}^{n-1} x_k y_{j-k \bmod n}. \tag{23}$$

For pairs the formula (23) reads explicitly

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \circledast \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \;=\; \begin{pmatrix} x_0 y_0 + x_1 y_1 \\ x_0 y_1 + x_1 y_0 \end{pmatrix}. \tag{24}$$

Let us note that if we tried to interpret (24) in terms of *vectors* we would have to implicitly assume that the pairs on both sides of (24) correspond to the *same* basis (otherwise the formula would be completely ambiguous). So let us take two different bases $\{\boldsymbol{b}_0, \boldsymbol{b}_1\}$ and $\{\boldsymbol{b}'_0, \boldsymbol{b}'_1\}$, and two vectors $\boldsymbol{x}$, $\boldsymbol{y}$. Each of these vectors can be written in both bases:

$$\boldsymbol{x} \;=\; x_0 \boldsymbol{b}_0 + x_1 \boldsymbol{b}_1 = x'_0 \boldsymbol{b}'_0 + x'_1 \boldsymbol{b}'_1, \tag{25}$$
$$\boldsymbol{y} \;=\; y_0 \boldsymbol{b}_0 + y_1 \boldsymbol{b}_1 = y'_0 \boldsymbol{b}'_0 + y'_1 \boldsymbol{b}'_1. \tag{26}$$

Circular convolutions of vectors would be meaningful if in any two bases we would find

$$\boldsymbol{x} \circledast \boldsymbol{y} \;=\; (x_0 y_0 + x_1 y_1)\boldsymbol{b}_0 + (x_0 y_1 + x_1 y_0)\boldsymbol{b}_1 = (x'_0 y'_0 + x'_1 y'_1)\boldsymbol{b}'_0 + (x'_0 y'_1 + x'_1 y'_0)\boldsymbol{b}'_1 \tag{27}$$

which is not the case. Indeed, let us take the basis rotated by $\pi/2$ ($\boldsymbol{b}'_0 = \boldsymbol{b}_1$, $\boldsymbol{b}'_1 = -\boldsymbol{b}_0$, $x'_0 = x_1$, $x'_1 = -x_0$, $y'_0 = y_1$, $y'_1 = -y_0$)

$$\boldsymbol{x} \;=\; x'_0 \boldsymbol{b}'_0 + x'_1 \boldsymbol{b}'_1 = x_1 \boldsymbol{b}_1 + (-x_0)(-\boldsymbol{b}_0), \tag{28}$$
$$\boldsymbol{y} \;=\; y'_0 \boldsymbol{b}'_0 + y'_1 \boldsymbol{b}'_1 = y_1 \boldsymbol{b}_1 + (-y_0)(-\boldsymbol{b}_0), \tag{29}$$

implying

$$(x'_0 y'_0 + x'_1 y'_1)\boldsymbol{b}'_0 + (x'_0 y'_1 + x'_1 y'_0)\boldsymbol{b}'_1 \;=\; (x_1 y_0 + x_0 y_1)\boldsymbol{b}_0 + (x_0 y_0 + x_1 y_1)\boldsymbol{b}_1 \tag{30}$$
$$\neq\; (x_0 y_0 + x_1 y_1)\boldsymbol{b}_0 + (x_0 y_1 + x_1 y_0)\boldsymbol{b}_1. \tag{31}$$

In contrast, for the geometric product we find

$$\boldsymbol{x}\boldsymbol{y} \;=\; (x_0 y_0 + x_1 y_1)\mathbf{1} + (x_0 y_1 - x_1 y_0)\boldsymbol{b}_0 \boldsymbol{b}_1 = (x'_0 y'_0 + x'_1 y'_1)\mathbf{1} + (x'_0 y'_1 - x'_1 y'_0)\boldsymbol{b}'_0 \boldsymbol{b}'_1, \tag{32}$$

which does not depend on the choice of basis. Let us note that the difference between geometric product and circular convolution boils down in this example to a single change of sign and reshuffling of components. To understand the

latter property we have to bear in mind that the GA of a 2D plane is 2^2-dimensional. A general multivector is here of the form $\psi = \alpha\mathbf{1} + \beta\boldsymbol{b}_0 + \gamma\boldsymbol{b}_1 + \delta\boldsymbol{b}_0\boldsymbol{b}_1$. Our calculation, in terms of the 4-tuples $(\alpha, \beta, \gamma, \delta)$, reads

$$\begin{pmatrix} 0 \\ x_0 \\ x_1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ y_0 \\ y_1 \\ 0 \end{pmatrix} = \begin{pmatrix} x_0y_0 + x_1y_1 \\ 0 \\ 0 \\ x_0y_1 - x_1y_0 \end{pmatrix}. \tag{33}$$

There do exist matrix representations of GA (cf. Section IX). It is known, however, that matrix representations of GA are not the most efficient implementations of GA-based algorithms. The algorithms that work efficiently in practice are based on direct calculations performed in terms of the GA rules (cf. [43]).

IV. CIRCULAR CONVOLUTION IN THE FOURIER SPACE VS. MATRIX ALGEBRA

A general multivector (22) can be represented by the 2^n-tuple $(\psi_{0_1\ldots0_n}, \ldots, \psi_{1_1\ldots1_n})$. The neutral element $\mathbf{1}$ corresponds in this notation to $(1, 0, \ldots, 0)$.

Similarly, the neutral element of the $\circledast$-algebra of n-tuples is the n-tuple $I = (1, 0, \ldots, 0)$. By definition, the $\circledast$-inverse x^{-1} of the n-tuple x satisfies $x^{-1} \circledast x = I$. An important map is the 'involution'

$$(x^*)_j = x_{-j \bmod n}. \tag{34}$$

Let us now rewrite $\circledast$ and $*$ in the Fourier space. The Fourier transform of $(x_0, \ldots, x_{n-1})$,

$$\hat{x}_k \;=\; \sum_{l=0}^{n-1} x_l e^{-2\pi ikl/n}, \tag{35}$$

satisfies

$$\widehat{(x \circledast y)}_k \;=\; \hat{x}_k\hat{y}_k, \tag{36}$$

$$\widehat{(x^*)}_k \;=\; \overline{\hat{x}_k}, \tag{37}$$

$$\widehat{(x^* \circledast y)}_k \;=\; \overline{\hat{x}_k}\hat{y}_k, \tag{38}$$

where $\overline{\hat{x}_k}$ denotes complex conjugation. The Fourier transform of $I = (1, 0, \ldots, 0)$ is $\hat{I} = (1, 1, \ldots, 1)$. Thus

$$\widehat{(x^{-1})}_k \;=\; \frac{1}{\hat{x}_k}. \tag{39}$$

x is not $\circledast$-invertible if any component of its Fourier transform is 0. This is why exact inverses are not used in standard HRR. Plate explains in [12] why x^* may be regarded as an approximate inverse of x, and why in the presence of noise — a generic situation in HRR — application of exact inverses would lead to unstable algorithms. Only in the case of unitary x (i.e. such that $x^* = x^{-1}$, $|\hat{x}_k| = 1$) the approximate inverse is exact. In GA, in contrast to HRR, exact inverses of nonzero vectors always exist, do not lead to instabilities, and are easy to calculate since $x^{-1} = x/|x|^2$.

Circular convolution in the Fourier space is thus equivalent to multiplication of diagonal matrices. Accordingly, the $\circledast$-inverse is the matrix inverse, and involution means Hermitian conjugation. The notion of $\circledast$-unitarity coincides with matrix unitarity. Fourier-space HRR involves binding represented by the matrix product of diagonal matrices and superposition is performed by matrix addition. HRR is implicitly an operator procedure but involving only commuting operators. These operators are in general neither Hermitian nor unitary but have the property of being *normal*, i.e. commute with their Hermitian conjugates.

Plate mentions in [12] (Section 3.6.7) that commutativity can cause ambiguities, so certain noncommutative variants of $\circledast$ may be in principle considered. For example, combination of $\circledast$ with permutations of components introduces noncommutativity for the price of associativity. Still another alternative mentioned in [12] is to work with vectors that can be written as matrices (i.e. with $n = m^2$, for some m) and use matrix multiplication. Apparently, this kind of reduced representation has not been studied in the literature so far.

Our claim is that the GA formalism is a natural noncommutative alternative to HRR, but to appreciate it we have to go deeper into the structure of the Fourier-space HRR.

V. FROM FOURIER-SPACE HRR TO BSC

Consider a general HRR-type Fourier-space superposition of N diagonal matrices U_j: $\psi = \sum_{j=1}^{N} U_j$. Now let us restrict U_j to matrices of the form $U_j = e^{i\pi P_{x_j}} = (-1)^{P_{x_j}}$ where P_{x_j} are diagonal matrices whose only nonzero elements are equal to 1. In other words, a diagonal of P_{x_j} is a sequence of bits: $P_{x_j} = \mathrm{diag}(x_{j,1}, \ldots, x_{j,n})$, $x_{j,k} = 0, 1$. Such a P_{x_j} is a projector: $P_{x_j}^2 = P_{x_j}$. Under these restrictions the diagonal unitary matrix U_j has the diagonal consisting of $(-1)^{x_{j,k}} = \pm 1$. The next step is to consider the new diagonal matrix $\Psi = \mathrm{sign}(\psi)$ defined via the spectral theorem from

$$\mathrm{sign}(x) \;=\; \begin{cases} +1 & \text{for } x \geq 0 \\ -1 & \text{for } x < 0 \end{cases} . \tag{40}$$

Ψ is again a unitary diagonal matrix whose only nonzero elements are equal to ± 1 and hence can be written as $\Psi = e^{i\pi P_x} = (-1)^{P_x}$. $P_x = \mathrm{diag}(x_1, \ldots, x_n)$ is a new projector, i.e. a diagonal matrix with bits on the diagonal. In effect, we have produced a new binary sequence $(x_1, \ldots, x_n)$ from a collection of N binary sequences $(x_{j,1}, \ldots, x_{j,n})$. The relevant formula (majority-rule summation) reads

$$x_k \;=\; \boxplus_{j=1}^{N} x_{j,k} = \Theta\Big(\frac{1}{N}\sum_{j=1}^{N} x_{j,k} - \frac{1}{2}\Big) \tag{41}$$

where

$$\Theta(x) \;=\; \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{42}$$

is the Heaviside step function.

The final step leading to BSC is to treat each U_j as a product of two unitary diagonal matrices R_j and F_j, also consisting of pluses and minuses on diagonals,

$$\Psi \;=\; \mathrm{sign}(\psi) = (-1)^{P_x} = \mathrm{sign}\Big(\sum_{j=1}^{N} R_j F_j\Big) = \mathrm{sign}\Big(\sum_{j=1}^{N} (-1)^{P_{x_j}} (-1)^{P_{y_j}}\Big) = \mathrm{sign}\Big(\sum_{j=1}^{N} (-1)^{(P_{x_j} \oplus P_{y_j})}\Big). \tag{43}$$

R_j, P_{x_j} represent roles, F_j, P_{y_j} are fillers, and $P_{x_j} \oplus P_{y_j}$ denotes matrix addition mod 2.

Since P_{x_j} and P_{y_j} are diagonal matrices with 0s and 1s on the diagonals, say, $P_{x_j} = \mathrm{diag}(x_{j,1}, \ldots, x_{j,n})$, $P_{y_j} = \mathrm{diag}(y_{j,1}, \ldots, y_{j,n})$, the map

$$(x_1, \ldots, x_n) \;=\; \boxplus_{j=1}^{N} (x_{j,1}, \ldots, x_{j,n}) \oplus (y_{j,1}, \ldots, y_{j,n}) \tag{44}$$

is the thresholded majority-rule componentwise addition of n-dimensional sums mod 2 (n-dimensional exclusive alternatives, XORs).

Now consider a single bit X and a sequence of N bits $(x_1, \ldots, x_N)$. Then the following distributivity of $\oplus$ over $\boxplus$ holds true:

$$X \oplus \big(\boxplus_{j=1}^{N} x_j\big) \;=\; \boxplus_{j=1}^{N} \big(X \oplus x_j\big). \tag{45}$$

(45) naturally generalizes to the n-dimensional variants of $\boxplus$ and XOR. Eq. (44) is the BSC where binary strings are bound by $\oplus$ and superposed by thresholded addition. Decoding of information is based in BSC on (45).

Let us rephrase the main result as follows. Circular convolution of n-tuples whose entries consist of ± 1 may be regarded as a multiplicative representation of XOR of n-bit strings, and the appropriate map is

$$x \oplus y \mapsto (-1)^{(P_x \oplus P_y)} = (-1)^{P_x} (-1)^{P_y} = RF. \tag{46}$$

The link between HRR and BSC is given by the map $x \mapsto (-1)^{P_x}$, where x on its left-hand side is a binary string of length n while on the right-hand side x occurs at the diagonal of an $n \times n$ matrix P_x.

In the next Section we will see that geometric product represents the *same* structure but in a *projective* way.

VI. GEOMETRIC PRODUCT AS A PROJECTIVE REPRESENTATION OF XOR

Let $x_1 \dots x_n$ and $y_1 \dots y_n$ be binary representations of two n-bit numbers x and y. Now let us consider two blades $c_x = c_{x_1 \dots x_n} = b_1^{x_1} \dots b_n^{x_n}$, $c_y = c_{y_1 \dots y_n} = b_1^{y_1} \dots b_n^{y_n}$. We will show that geometric product of c_x and c_y equals, up to a sign, $c_{x \oplus y}$. In this sense the map $x \mapsto c_x$ is an analogue of the exponential map $x \mapsto (-1)^{P_x}$.

Let us begin with examples:

$$\begin{aligned} b_1 b_1 &= c_{10\dots0} c_{10\dots0} = 1 = c_{0\dots0} = c_{(10\dots0)\oplus(10\dots0)} && (47)\\ b_1 b_{12} &= c_{10\dots0} c_{110\dots0} = b_1 b_1 b_2 = b_2 = c_{010\dots0} = c_{(10\dots0)\oplus(110\dots0)} && (48)\\ b_{12} b_1 &= c_{110\dots0} c_{10\dots0} = b_1 b_2 b_1 = -b_2 b_1 b_1 = -b_2 = -c_{010\dots0} = -c_{(110\dots0)\oplus(10\dots0)} && (49)\\ b_{1257} b_{26} &= c_{11001010\dots0} c_{0100010\dots0} = b_1 b_2 b_5 b_7 b_2 b_6 \\ &= (-1)^3 b_1 b_5 b_6 b_7 = (-1)^3 c_{10001110\dots0} = (-1)^D c_{(11001010\dots0)\oplus(0100010\dots0)}. && (50) \end{aligned}$$

The number D is the number of times a 1 from the right string had to "jump" over a 1 from the left one during the process of shifting the right string to the left.

The above observations, generalized to arbitrary strings of bits, yield

$$c_{x_1 \dots x_n} c_{y_1 \dots y_n} = (-1)^{\sum_{k<l} y_k x_l} c_{(x_1 \dots x_n)\oplus(y_1 \dots y_n)}. \quad (51)$$

Indeed, for two arbitrary strings of bits we have

$$D = y_1(x_2 + \dots + x_n) + y_2(x_3 + \dots + x_n) + \dots + y_{n-1} x_n = \sum_{k<l} y_k x_l. \quad (52)$$

We conclude that the map

$$(x_1, \dots, x_n) \times (y_1, \dots, y_n) \mapsto (x_1, \dots, x_n) \oplus (y_1, \dots, y_n) = (x_1 \oplus y_1, \dots, x_n \oplus y_n) \quad (53)$$

is projectively (i.e. up to a sign) represented in GA by means of (51).

Representations 'up to a sign' play an important role in physics and are at the roots of many phenomena such as half-integer spin and fermionic statistics. To the best of our knowledge the link between projective representations of XOR and GA was noticed for the first time in the preliminary version of this paper [44]

VII. GEOMETRIC ANALOGUE OF HRR

Let us begin with illustrating the original BSC by means of the example taken from [14]. The records are represented by unstructured randomly chosen strings of bits. The encoded record is

$$\mathbf{PSmith} = (\mathbf{name} \oplus \mathbf{Pat}) \boxplus (\mathbf{sex} \oplus \mathbf{male}) \boxplus (\mathbf{age} \oplus \mathbf{66}). \quad (54)$$

Decoding of the "name" looks as follows

$$\begin{aligned} \mathbf{Pat}' &= \mathbf{name} \oplus \mathbf{PSmith} \\ &= \mathbf{name} \oplus \left[(\mathbf{name} \oplus \mathbf{Pat}) \boxplus (\mathbf{sex} \oplus \mathbf{male}) \boxplus (\mathbf{age} \oplus \mathbf{66})\right] \\ &= \mathbf{Pat} \boxplus (\mathbf{name} \oplus \mathbf{sex} \oplus \mathbf{male}) \boxplus (\mathbf{name} \oplus \mathbf{age} \oplus \mathbf{66}) \\ &= \mathbf{Pat} \boxplus \text{noise} \to \mathbf{Pat}. \end{aligned} \quad (55)$$

We have used here the involutive nature of XOR and the fact that the "noise" can be eliminated by clean-up memory. The latter means that we compare $\mathbf{Pat}'$ with records stored in some memory and check, by means of the Hamming distance, which of the stored elements is closest to $\mathbf{Pat}'$. A similar trick could be done be means of circular convolution in HRRs, but then we would have used the inverse $\mathbf{name}^{-1}$ or the involution $\mathbf{name}^*$, and an appropriate measure of distance. Again, the last step is comparison of the noisy object with the "pure" objects stored in clean-up memory. This is how standard BSC works.

We can now use the exponential map $x \mapsto (-1)^{P_x}$ to turn BSC into HRR. Let us, however, proceed in the geometric way and employ $x \mapsto c_x$. The roles and fillers are represented by randomly chosen blades:

$$\mathbf{PSmith} = \mathbf{name} \cdot \mathbf{Pat} + \mathbf{sex} \cdot \mathbf{male} + \mathbf{age} \cdot \mathbf{66}. \quad (56)$$

The dot "$\cdot$" is the geometric product and $\boxplus$ is replaced, similarly to HRR, by ordinary addition. At the level of explicit blades the record corresponds to the multivector **PSmith**

$$\mathbf{PSmith} = c_{a_1\ldots a_n}c_{x_1\ldots x_n} + c_{b_1\ldots b_n}c_{y_1\ldots y_n} + c_{c_1\ldots c_n}c_{z_1\ldots z_n}. \tag{57}$$

The blades indexed by the beginning of the alphabet represent roles (name, sex, age) while the remaining ones correspond to the fillers (Pat, male, 66). The decoding looks as follows

$$\begin{aligned}\mathbf{name}\cdot\mathbf{PSmith} &= c_{a_1\ldots a_n}\left[c_{a_1\ldots a_n}c_{x_1\ldots x_n} + c_{b_1\ldots b_n}c_{y_1\ldots y_n} + c_{c_1\ldots c_n}c_{z_1\ldots z_n}\right] &(58)\\ &= \pm c_x \pm c_{a\oplus b\oplus y} \pm c_{a\oplus c\oplus z} &(59)\\ &= \pm\mathbf{Pat} + \text{noise}. &(60)\end{aligned}$$

The signs have to computed by means of (51).

An analogue of clean-up memory can be constructed in various ways. One possibility is to make sure that fillers, c_x etc. are orthogonal to the noise term. For example, let us take the fillers of the form $c_{x_1\ldots x_k 0\ldots 0}$, where the first $k \ll n$ bits are selected at random, but the remaining $n-k$ bits are all 0. Let the roles be taken, as in Kanerva's BSC, with *all* the bits generated at random. The term $c_{(x_1\ldots x_n)\oplus(y_1\ldots y_n)\oplus(y_1\ldots y_n)}$ will with high probability contain at least one $y_j = 1$, $k < j \leq n$, and thus will be orthogonal to the fillers. The clean-up memory will consist of vectors with $y_j = 0$, $k < j \leq n$, i.e of the filler form.

VIII. BINARY CODING IN DIFFERENT BASES

In quantum mechanics bits can be associated with any set of qubits (i.e. with any basis in a 2-dimensional complex space). The freedom to choose the basis is crucial for quantum cryptography, but is generally not used in QC. In QC one typically works in the so-called computational basis, which is fixed in advance. This, of course, does not change the fact that the whole formalism of QC is based on vectors and not on n-tuples.

A similar situation occurs in our GA analogue of HRR. A multivector ψ (22) is a combination of blades, and blades represent binary numbers only when we fix the basis. To put it differently, the same single multivector ψ can be associated with different reduced representations, but whether this freedom is of any practical use is an open question.

IX. CARTAN REPRESENTATION OF CLIFFORD ALGEBRAS

It is useful to be able to work with matrix representations of GA. Although this is not an efficient way of doing GA computations, matrix representations allow to perform independent cross-checks of various GA constructions and algorithms. In this section we give an explicit matrix representation of GA. We begin with Pauli's matrices

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{61}$$

GA of a plane is represented as follows: $1 = 2\times 2$ unit matrix, $b_1 = \sigma_1$, $b_2 = \sigma_2$, $b_{12} = \sigma_1\sigma_2 = i\sigma_3$. Alternatively, we can write $c_{00} = 1$, $c_{10} = \sigma_1$, $c_{01} = \sigma_2$, $c_{11} = i\sigma_3$, and

$$\psi_{00}c_{00} + \psi_{10}c_{10} + \psi_{01}c_{01} + \psi_{11}c_{11} = \begin{pmatrix} \psi_{00} + i\psi_{11} & \psi_{10} - i\psi_{01} \\ \psi_{10} + i\psi_{01} & \psi_{00} - i\psi_{11} \end{pmatrix}. \tag{62}$$

This is equivalent to encoding $2^2 = 4$ real numbers into two complex numbers.

In 3-dimensional space we have $1 = 2\times 2$ unit matrix, $b_1 = \sigma_1$, $b_2 = \sigma_2$, $b_3 = \sigma_3$, $b_{12} = \sigma_1\sigma_2 = i\sigma_3$, $b_{13} = \sigma_1\sigma_3 = -i\sigma_2$, $b_{23} = \sigma_2\sigma_3 = i\sigma_1$, $b_{123} = \sigma_1\sigma_2\sigma_3 = i$.

Now the representation of

$$\sum_{ABC=0,1}\psi_{ABC}c_{ABC} = \begin{pmatrix} \psi_{000} + i\psi_{111} + \psi_{001} + i\psi_{110}, & \psi_{100} + i\psi_{011} - i\psi_{010} - \psi_{101} \\ \psi_{100} + i\psi_{011} + i\psi_{010} + \psi_{101}, & \psi_{000} + i\psi_{111} - \psi_{001} - i\psi_{110} \end{pmatrix} \tag{63}$$

is equivalent to encoding $2^3 = 8$ real numbers into 4 complex numbers.

An arbitrary n-bit record can be encoded into the matrix algebra known as Cartan's representation of Clifford algebras [45]:

$$b_{2k} = \underbrace{\sigma_1 \otimes \cdots \otimes \sigma_1}_{n-k} \otimes \sigma_2 \otimes \underbrace{1 \otimes \cdots \otimes 1}_{k-1}, \tag{64}$$

$$b_{2k-1} = \underbrace{\sigma_1 \otimes \cdots \otimes \sigma_1}_{n-k} \otimes \sigma_3 \otimes \underbrace{1 \otimes \cdots \otimes 1}_{k-1}. \tag{65}$$

So let us return to the example of Pat Smith. For simplicity take $n = 4$ so that we can choose the representation

$$\left.\begin{aligned} \mathbf{Pat} &= c_{1100}, \\ \mathbf{male} &= c_{1000}, \\ \mathbf{66} &= c_{0100}, \end{aligned}\right\} \text{fillers} \tag{66}$$

$$\left.\begin{aligned} \mathbf{name} &= c_{1010}, \\ \mathbf{sex} &= c_{0111}, \\ \mathbf{age} &= c_{1011}. \end{aligned}\right\} \text{roles} \tag{67}$$

The fillers have only the first two bits selected at random, the last two are 00. The roles are numbered by randomly selected strings of bits.

The explicit matrix representations are:

$$\mathbf{Pat} = c_{1100} = b_1 b_2 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2) = 1 \otimes 1 \otimes 1 \otimes (-i\sigma_1) \tag{68}$$

$$\mathbf{male} = c_{1000} = b_1 = \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3 \tag{69}$$

$$\mathbf{66} = c_{0100} = b_2 = \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2 \tag{70}$$

$$\begin{aligned} \mathbf{name} &= c_{1010} = b_1 b_3 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1) \\ &= 1 \otimes 1 \otimes (-i\sigma_2) \otimes \sigma_3 \\ \mathbf{sex} &= c_{0111} = b_2 b_3 b_4 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1)(\sigma_1 \otimes \sigma_1 \otimes \sigma_2 \otimes 1) \\ &= \sigma_1 \otimes \sigma_1 \otimes (-i1) \otimes \sigma_2, \end{aligned} \tag{71}$$

$$\begin{aligned} \mathbf{age} &= c_{1011} = b_1 b_3 b_4 = (\sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_3)(\sigma_1 \otimes \sigma_1 \otimes \sigma_3 \otimes 1)(\sigma_1 \otimes \sigma_1 \otimes \sigma_2 \otimes 1) \\ &= \sigma_1 \otimes \sigma_1 \otimes (-i1) \otimes \sigma_3 \end{aligned} \tag{72}$$

The whole record — where the superposition is taken for clarity with arbitrary parameters α, β, and γ — reads

$$\begin{aligned} \mathbf{PSmith} &= \alpha\, \mathbf{name} \cdot \mathbf{Pat} + \beta\, \mathbf{sex} \cdot \mathbf{male} + \gamma\, \mathbf{age} \cdot \mathbf{66} \\ &= \alpha c_{1010} c_{1100} + \beta c_{0111} c_{1000} + \gamma c_{1011} c_{0100} \\ &= \alpha c_{0110} - \beta c_{1111} + \gamma c_{1111} \end{aligned}$$

The two noise terms are here linearly dependent by accident. This is a consequence of too small dimensionality of our binary strings (four bits, whereas in realistic cases Kanerva suggested 10^4 bit strings). This is the price we pay for simplicity of the example. Decoding the name involves two steps. First

$$\begin{aligned} \mathbf{name} \cdot \mathbf{PSmith} &= c_{1010} \cdot \mathbf{PSmith} = c_{1010}\big[\alpha c_{0110} - (\beta - \gamma) c_{1111}\big] \\ &= -\alpha c_{1100} - (\beta - \gamma) c_{0101} \\ &= -\alpha \mathbf{Pat} \underbrace{-(\beta - \gamma) c_{0101}}_{\text{noise}} = -\alpha \underbrace{b_1 b_2}_{\text{Pat}} \underbrace{-(\beta - \gamma) b_2 b_4}_{\text{noise}} = \mathbf{Pat}'. \end{aligned} \tag{73}$$

It remains to employ clean-up memory. But this is easy since the noise is perpendicular to **Pat**. We only have to project on the set spanned by the fillers (i.e. the blades involving neither b_3 nor b_4), and within this set check which element is closest to the cleaned up $\mathbf{Pat}'$.

X. CONCLUSIONS

In order to switch from a binary string x, occurring in BSC, to HRR, one employs the exponential map $x \mapsto (-1)^{P_x}$ where x plays a dual role. At the left-hand side x is just a sequence of bits distributed over a n-tuple. At the

right-hand side the bits are distributed over the diagonal of a diagonal $n \times n$ matrix P_x. Binary strings equipped with componentwise addition mod 2 (i.e. $\oplus$, XOR) form a group. The exponential map is a representation of this group in the space of diagonal matrices: $x \oplus y \mapsto (-1)^{P_x \oplus P_y} = (-1)^{P_x}(-1)^{P_y}$.

This group possesses also a projective representation in the space of blades of a GA: $x \oplus y \mapsto c_{x\oplus y} = \pm c_x c_y$. Blades have a straightforward geometric interpretation. As opposed to tensor products, that increase dimensions, the dimensions of c_x, c_y and $c_{x\oplus y}$ are the same. Therefore, $c_{x\oplus y}$ can be used to bind variables. A superposition of such bound variables, a multivector, is a reduced representation analogous to HRR.

GA may also be interpreted as a way of encoding mutual geometric relations between multidimensional geometric objects. For example, a pair containing an oriented plane element and a vector from this plane is mapped in GA into a vector which shows how to move the first vector in order to produce the oriented plane segment in question. The algebraic operation thus reveals a geometric property of the plane segment, and resembles the process of *understanding* geometry.

We find the latter observation at least intriguing. It suggests that association of GA with cognitive science is not just a mathematical curiosity, but may be deeply rooted in the ways we think.

Acknowledgments

This work was supported by the Flemish Fund for Scientific Research (FWO), project G.0452.04.

[1] G. E. Hinton, Mapping part-whole hierarchies into connectionist networks, Artificial Intelligence **46**, 47-76 (1990).
[2] P. Smolensky, Tensor product variable binding and the representation of symbolic structures in connectionist systems, Artificial Intelligence **46**, 159-216 (1990).
[3] D. Gabor, Holographic model for temporal recall, Nature **217**, 1288-1289 (1968).
[4] D. Willshaw, Holography, associative memory, and inductive generalization, in *Parallel Models of Associative Memory (updated edition)*, G. E. Hinton and J. A. Anderson (eds.) (Hillsdale, Erlbaum, 1989) 103–124
[5] A. Borsellino and T. Poggio, Convolution and correlation algebras, Kybernetik **13**, 113-122 (1973).
[6] P. Liepa, Models of content addressable distributed associative memory, unpublished manuscript (1977).
[7] B. B. Murdock, A theory for the storage and retrieval of item and associative information, Psychological Review **89**, 316–338 (1982).
[8] J. Metcalfe, Recognition failure and CHARM, Psychological Review **98**, 529–553 (1991)
[9] J. Metcalfe-Eich, A composite holographic associative recall model, Psychological Review **89**, 627–661 (1982)
[10] J. N. Slack, The role of distributed memory in natural language processing, in *Advances in Artificial Intelligence: Proceedings of the 6th European Conference on Artificial Intelligence, ECAI-84*, T. O'Shea (ed.) (Elsevier, 1984).
[11] T. Plate, Holographic reduced representations, IEEE Transactions on Neural Networks **6**, 623-641 (1995).
[12] T. Plate, *Holographic Reduced Representation: Distributed Representation for Cognitive Structures* (CSLI Publications, Stanford, 2003).
[13] P. Kanerva, Binary spatter codes of ordered k-tuples, *Artificial Neural Networks–ICANN Proceedings*, Lecture Notes in Computer Science vol. 1112, pp. 869-873, C. von der Malsburg *et al.* (Eds.) (Springer, Berlin, 1996).
[14] P. Kanerva, Fully distributed representation, *Proc. 1997 Real World Computing Symposium (RWC'97, Tokyo)*, pp. 358-365 (Real World Computing Partnership, Tsukuba-City, Japan, 1997).
[15] P. Kanerva, Large patterns make great symbols: An example of learning from example, *Hybrid Neural Systems*, pp. 194-203 (1998).
[16] M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).
[17] D. Aerts and M. Czachor, Quantum aspects of semantic analysis and symbolic artificial intelligence, Journal of Physics A **37**, L123-L132 (2004).
[18] D. Widdows, *Geometry and Meaning* (CSLI Publications, Stanford, 2004).
[19] D. Widdows and M. Higgins, Geometric ordering of concepts, logical disjunction, and learning by induction, *Compositional Connectionism in Cognitive Science*, AAAI Fall Symposium Series, Washington, DC, October 22-24, 2004.
[20] R. Penrose, *Shadows of the Mind* (Oxford University Press, Oxford, 1994).
[21] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought* (Oxford University Press, Oxford, 2003).
[22] L. Dorst, D. Fontijne, and S. Mann, *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*, The Morgan Kaufmann Series in Computer Graphics (Morgan Kaufmann, Amsterdam, 2007).
[23] D. Aerts and M. Czachor, Cartoon computation: Quantum-like algorithms without quantum mechanics, J. Phys. A **40**, F259 (2007).
[24] M. Czachor, Elementary gates for cartoon computation, J. Phys. A **40**, F753 (2007).

[25] D. Aerts and M. Czachor, Tensor-product vs. geometric-product coding, preprint arXiv:0709.1268 [quant-ph] (2007) — submited to Physical Review A.
[26] S. Somaroo, D. G. Cory, and T. F. Havel, Expressing the operations of quantum computing in multiparticle geometric algebra, Physics Letters A **240**, 1-7 (1998).
[27] H. Grassmann, Der Ort der Hamilton'schen Quaternionen in der Ausdehnungslehre, Mathematische Annalen **3**, 375–386 (1877).
[28] W. K. Clifford, Applications of Grassmann's extensive algebra, American Journal of Mathematics Pure and Applied **1**, 350–358 (1878).
[29] D. Hestenes, *Space-Time Algebra* (Gordon and Breach, New York, 1966).
[30] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics* (Reidel, Dordrecht, 1984).
[31] D. Hestenes, *New Foundations for Classical Mechanics* (Kluwer, Dordrecht, 1986).
[32] D. Hestenes, Reforming the mathematical language of physics, American Journal of Physics **71**, 104-121 (2003).
[33] W. E. Baylis, *Electrodynamics: A Modern Geometric Approach* (Birkhauser, Boston, 1996).
[34] G. Sommer (ed.), *Geometric Computing with Clifford Algebras* (Springer, Berlin 2001).
[35] L. Dorst, C. J. L. Doran, J. Lasenby (eds.), *Applications of Geometric Algebra in Computer Science and Engineering* (Birkhauser, Boston, 2002).
[36] M. Pavsic, *The Landscape of Theoretical Physics: A Global View. From Point Particles to the Brane World and Beyond, in Search of a Unifying Principle* (Kluwer, Boston, 2001).
[37] C. Doran and A. Lasenby, *Geometric Algebra forPhysicists* (Cambridge University Press, Cambridge, 2003).
[38] J. Lasenby, A. N. Lasenby, C. J. L. Doran, and W. J. Fitzgerald, New geometric methods for computer vision, International Journal of Computer Vision **36**, 191-213 (1998).
[39] E. Bayro-Corrochano, K. Danilidis, and G. Sommer, Motor algebra for 3D kinematics: The case of the hand-eye calibration, Journal of Mathematical Imaging and Vision **13**, 79-100 (2000).
[40] E. J. Bayro-Corrochano, Geometric neural computing, IEEE Transactions on Neural Networks **12**, 968-986 (2001).
[41] W. Gibbs, The Scientific Papers of William Gibbs, vol. III (Longmas, Green and Company, London, 1906).
[42] One can start, say, with http://www.lomont.org/Math/GeometricAlgebra/Papers.php, and browse through the links.
[43] GABLE: Geometric AlgeBra Learning Environment, http://staff.science.uva.nl/∼leo/clifford/gable.htm; GAIGEN: Geometric Algebra Implementation Generator, http://sourceforge.net/projects/geigen; GAViewer: Interactive Geometric Algebra with OpenGL Visualization, http://www.science.uva.nl/ga/viewer/; CLUCalc, http://www.perwass.de/CLU/index.html; GA Package for Maple, http://www.mrao.cam.ac.uk/∼maja1/software/GA/; CLIFFORD — A Maple Package for Clifford Algebra Computations with Bigebra, http://math.tntech.edu/rafal/
[44] D. Aerts, M. Czachor, and B. De Moor, On geometric-algebra representation of binary spatter codes, preprint arXiv:cs/0610075 [cs.AI] (2006).
[45] P. Budinich and A. Trautman, *The Spinorial Chessboard* (Springer, Berlin, 1988).
[46] A complex structure, if needed, may be introduced by means of an additional bit without any need of complex numbers. This is how the 'imaginary unit' i was used in [24] in the context of quantum gates. It must be stressed, however, that in the GA literature it is a tradition to replace i by a blade. For example, we have seen that in 2D $(b_1 b_2)^2 = -\mathbf{1}$ which explains why $b_1 b_2$ has properties analogous to i. Nevertheless, this construction does not properly work in our context (cf. the discussion in [25]) and we have to use a different representation of i — equivalent to a $\pi/2$ rotation in 2D. These subtleties are not important for the discussion given in the present paper.