



Efficient representation and manipulation of quadratic surfaces using Geometric Algebras

Stéphane Breuils, Vincent Nozick, Laurent Fuchs, Akihiro Sugimoto

► To cite this version:

Stéphane Breuils, Vincent Nozick, Laurent Fuchs, Akihiro Sugimoto. Efficient representation and manipulation of quadratic surfaces using Geometric Algebras. 2019. hal-02050767v1

HAL Id: hal-02050767

<https://hal.science/hal-02050767v1>

Preprint submitted on 27 Feb 2019 (v1), last revised 1 Apr 2019 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Efficient representation and manipulation of quadratic surfaces using Geometric Algebras

Stéphane Breuils¹, Vincent Nozick¹, Laurent Fuchs² and Akihiro Sugimoto³

¹ Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI,
UMR 8049, Université Paris-Est Marne-la-Vallée, France
France

`firstName.name@u-pem.fr`

² XLIM-ASALI, UMR 7252,
Université de Poitiers, Poitiers, France
`Laurent.Fuchs@univ-poitiers.fr`

³ National Institute of Informatics,
Tokyo 101-8430, Japan,
`sugimoto@nii.ac.jp`

Abstract. Quadratic surfaces gain more and more attention among the Geometric Algebra community and some frameworks were proposed in order to represent, transform, and intersect these quadratic surfaces. As far as the authors know, none of these frameworks support all the operations required to completely handle these surfaces. Some frameworks do not allow the construction of quadratic surfaces from control points when others do not allow to transform these quadratic surfaces. However, if we consider all the frameworks together, then all the required operations over quadratic are covered. This paper presents a unification of these frameworks that enables to represent any quadratic surfaces either using control points or from the coefficients of its implicit form. The proposed approach also allows to transform any quadratic surfaces and to compute their intersection and to easily extract some geometric properties.

Keywords: Geometric Algebra, Quadratic surfaces, Conformal Geometric Algebras

1 Introduction

Geometric algebra provides convenient and intuitive tools to represent, transform, and intersect geometric objects. Deeply explored by physicists, it has been used in quantum mechanics and electromagnetism [12, 11] as well as in classical mechanics [13]. Geometric algebra has also found some interesting applications in geographic data manipulations [15, 21]. Among them, geometric algebra is used within the computer graphics community. More precisely, it is used not only in basis geometric primitive manipulations [19] but also in complex illumination processes as in [16] where spherical harmonics are substituted by

geometric algebra entities. Finally, in data and image analysis, we can find the usefulness of geometric algebra in mathematical morphology [6] and in neural networking [3, 14]. In the geometric algebra community, quadratic surfaces gain more and more attention, and some frameworks have been proposed in order to represent, transform, and intersect these quadratic surfaces.

There exist three main approaches to deal with quadratic surfaces. The first one, introduced in [8], is called double conformal geometric algebra of $G_{6,2}$. It is capable of representing quadratic surfaces from the coefficients of their implicit form. The second one is double perspective geometric algebra of $G_{4,4}$ whose definition was firstly introduced in [10]. It has been further developed in [7]. This approach is based on a duplication of \mathbb{R}^3 and also represents quadratic surfaces from the coefficients of their implicit form, as bivectors. To our best knowledge, it can not construct general quadrics from control points. The third one was introduced in [2] and is denoted as quadric conformal geometric algebra (QCGA). QCGA allows us to define general quadrics from 9 control points, and to represent the objects in low dimensional subspaces of the algebra. QCGA is capable of constructing quadratic surfaces either using control points or implicit equations as 1-vector. As far as we know, QCGA does not yet allow all geometric transformations of quadrics surfaces.

In order to enhance usefulness of Geometric Algebra for geometry and computer graphics community, a new framework that allows to represent and manipulate quadratic surfaces has to be developed. This is the main purpose of this paper.

All the examples and computations are based upon the efficient geometric algebra library generator Garamon [1]. The code of this library generator is available online⁴.

1.1 Contributions

We propose a new approach that unifies the three above framework. Using this approach, we show that it is possible to represent quadratic surfaces either using control points or its implicit coefficients but also transform these quadratic surfaces using versors. Computation of the tangent planes to a quadratic surfaces is also defined. We show that it is also possible to intersect quadratic surfaces in this new framework. Finally, this approach also enables to define also some quartic surfaces.

2 Notations:

Following the state-of-the-art usages of [5] and [18], upper-case bold letters denote blades (blade **A**) whose grade is higher than 1. Multivectors and k -vectors are denoted with upper-case non-bold letters (multivector *A*). Lower-case bold letters refer to vectors and lower-case non-bold to multivector coordinates. The

⁴ git clone <https://git.renater.fr/garamon.git>

k -grade part of a multivector A is denoted by $\langle A \rangle_k$. Finally, The vector space dimension is denoted by 2^d , where d is the number of basis blades e_i of grade 1.

3 Computational efficiency of models of Quadric surfaces with Geometric Algebras

3.1 Complexity estimation model

In order to compare the operations for these different frameworks, we need a computational model. This requires to be able to determine the complexity of operations in each framework. Using the complexity of the operations explained in the last part of this manuscript is well suited when one wants to compare different methods to compute the products. Instead, our complexity estimation is made through two simplifying assumptions.

First, let us consider the outer product between one homogeneous multivector whose number of components is u and another homogeneous multivector whose number of components is v , $u, v \in \mathbb{N}$. We assume that an upper bound to the number of required products is at most uv products, as shown in the definition of the outer product.

The equation of the inner product is our base for the second assumption. Furthermore, we need to use this formula for inner products between 1–vector and 2–vector as well as inner product between two 1–vector. The first multivector has u non-zero components, and the second has v non-zero components. Then the inner product between two 1–vectors will result in uv products. Whereas, the inner product between 1–vectors and 2-vectors requires two inner products for each pair of components of the two multivectors. Thus, this inner product requires at most $2uv$ products. These models will be used for determining the complexity of each operation. Let us now explain in more details these models, we start with DCGA.

3.2 Models of quadratic surfaces with Geometric Algebra

This section presents the main Geometric Algebra models to represent and manipulate quadratic surfaces. Furthermore, we aim at determining the most efficient Geometric Algebra model for each of the main operations required in computer graphics. The purpose is to propose a Geometric Algebra model that allows to efficiently handle quadrics.

The considered operations over the surfaces are:

- checking if a point lies in a quadratic surface,
- intersecting quadratic surface and line,
- computing the normal vector (and tangent plane) of a surface at a given point.

One of the applications of such operations is to compute precise visualisations using ray-tracer [9].

A first framework to handle quadratic surfaces was introduced thanks to the pioneering work of Zamora [20]. This framework constructs a quadratic surface from control points in $\mathbb{G}_{6,3}$. In this model, an axis-aligned quadratic surface \mathbf{Q} can be defined as:

$$\mathbf{Q} = \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 \wedge \mathbf{x}_4 \wedge \mathbf{x}_5 \wedge \mathbf{x}_6 \quad (1)$$

The major drawback of this model is that it supports only axis-aligned quadratic surfaces. Due to this fact, we will not further detail this model.

There exist three main Geometric Algebra frameworks to manipulate general quadratic surfaces. First, DCGA (Double Conformal Geometric Algebra) with $\mathbb{G}_{8,2}$ defined by Easter and Hitzer [8]. Second, a framework of $\mathbb{G}_{4,4}$ as firstly introduced by Parkin [17] and developed further by Du et al. [7]. The last one is our contribution, introduced in [2] and is a model of $\mathbb{G}_{9,6}$.

DCGA of $\mathbb{G}_{8,2}$ DCGA was presented by Hitzer and Easter [8] and aims at having entities representing both quartic surfaces and quadratic surfaces.

Basis and metric

In more details, the DCGA $\mathbb{G}_{8,2}$ is defined over a 10-dimensional vector space. The base vectors of the space are basically divided into two groups: $\{\mathbf{e}_{01}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_{\infty 1}\}$, corresponding to the CGA vectors defined in Chapter 1, and a copy of this basis $\{\mathbf{e}_{02}, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_{\infty 2}\}$. The inner products between them are defined in Table 1. Note that we highlighted in grey the two models of CGA that are included in DCGA.

Table 1: Inner product between DCGA basis vectors.

	\mathbf{e}_{01}	\mathbf{e}_1	\mathbf{e}_2	\mathbf{e}_3	$\mathbf{e}_{\infty 1}$	\mathbf{e}_{02}	\mathbf{e}_4	\mathbf{e}_5	\mathbf{e}_6	$\mathbf{e}_{\infty 2}$
\mathbf{e}_{01}	0	0	0	0	-1	0	0	0	0	0
\mathbf{e}_1	0	1	0	0	0	0	0	0	0	0
\mathbf{e}_2	0	0	1	0	0	0	0	0	0	0
\mathbf{e}_3	0	0	0	1	0	0	0	0	0	0
$\mathbf{e}_{\infty 1}$	-1	0	0	0	0	0	0	0	0	0
\mathbf{e}_{02}	0	0	0	0	0	0	0	0	0	-1
\mathbf{e}_4	0	0	0	0	0	0	1	0	0	0
\mathbf{e}_5	0	0	0	0	0	0	0	1	0	0
\mathbf{e}_6	0	0	0	0	0	0	0	0	1	0
$\mathbf{e}_{\infty 2}$	0	0	0	0	0	-1	0	0	0	0

Point of DCGA

A point of DCGA whose Euclidean coordinates are (x, y, z) is defined as the

outer product of two CGA points with coordinates (x, y, z) . By defining this two points \mathbf{x}_1 and \mathbf{x}_2 as:

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{e}_{o1} + x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3 + \frac{1}{2}(x^2 + y^2 + z^2)\mathbf{e}_{\infty 1} \\ \mathbf{x}_2 &= \mathbf{e}_{o2} + x\mathbf{e}_4 + y\mathbf{e}_5 + z\mathbf{e}_6 + \frac{1}{2}(x^2 + y^2 + z^2)\mathbf{e}_{\infty 2}\end{aligned}\quad (2)$$

The embedding of a DCGA point is defined as follows:

$$\mathbf{X} = \mathbf{x}_1 \wedge \mathbf{x}_2 \quad (3)$$

The development of this operation results in:

$$\begin{aligned}\mathbf{X} &= x^2\mathbf{e}_1 \wedge \mathbf{e}_4 + y^2\mathbf{e}_2 \wedge \mathbf{e}_5 + z^2\mathbf{e}_3 \wedge \mathbf{e}_6 + xy(\mathbf{e}_1 \wedge \mathbf{e}_5 + \mathbf{e}_2 \wedge \mathbf{e}_4) \\ &\quad + xz(\mathbf{e}_1 \wedge \mathbf{e}_6 + \mathbf{e}_3 \wedge \mathbf{e}_4) + yz(\mathbf{e}_2 \wedge \mathbf{e}_6 + \mathbf{e}_3 \wedge \mathbf{e}_5) \\ &\quad + x(\mathbf{e}_{o1} \wedge \mathbf{e}_4 + \mathbf{e}_1 \wedge \mathbf{e}_{o2}) + y(\mathbf{e}_{o1} \wedge \mathbf{e}_5 + \mathbf{e}_2 \wedge \mathbf{e}_{o2}) \\ &\quad + z(\mathbf{e}_{o1} \wedge \mathbf{e}_6 + \mathbf{e}_3 \wedge \mathbf{e}_{o2}) + \mathbf{e}_{o1} \wedge \mathbf{e}_{o2} \\ &\quad + \frac{1}{2}(x^2 + y^2 + z^2)(\mathbf{e}_{o1} \wedge \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1} \wedge \mathbf{e}_{o2}) \\ &\quad + \frac{1}{2}x(x^2 + y^2 + z^2)(\mathbf{e}_1 \wedge \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1} \wedge \mathbf{e}_4) \\ &\quad + \frac{1}{2}y(x^2 + y^2 + z^2)(\mathbf{e}_2 \wedge \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1} \wedge \mathbf{e}_5) \\ &\quad + \frac{1}{2}z(x^2 + y^2 + z^2)(\mathbf{e}_3 \wedge \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1} \wedge \mathbf{e}_6) \\ &\quad + \frac{1}{4}(x^2 + y^2 + z^2)^2\mathbf{e}_{\infty 1} \wedge \mathbf{e}_{\infty 2}\end{aligned}\quad (4)$$

This high number of components is due to the fact that the representation of a point of DCGA was designed to non only define quadratic surfaces but also quartic surfaces. To illustrate this, we highlight the components that contribute to construct quadrics in red. The other components are dedicated to the construction of quartics.

Quadratic surfaces

A general quadratic surface merely consists in defining some operators that extract the components of \mathbf{x} . For a general quadric defined as:

$$ax^2 + by^2 + cz^2 + dxy + eyz + fzx + gx + hy + iz + j = 0 \quad (5)$$

This means that 4 operators are defined for the quadratic part:

$$\begin{aligned}\mathbf{T}_{x^2} &= \mathbf{e}_4 \wedge \mathbf{e}_1 \quad \mathbf{T}_{y^2} = \mathbf{e}_5 \wedge \mathbf{e}_2 \\ \mathbf{T}_{z^2} &= \mathbf{e}_6 \wedge \mathbf{e}_3 \quad \mathbf{T}_1 = -\mathbf{e}_{\infty 1} \wedge \mathbf{e}_{\infty 2}\end{aligned}\quad (6)$$

along with the 3 operators for the linear part:

$$\begin{aligned}\mathbf{T}_x &= \frac{1}{2}(\mathbf{e}_1 \wedge \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1} \wedge \mathbf{e}_4) \\ \mathbf{T}_y &= \frac{1}{2}(\mathbf{e}_2 \wedge \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1} \wedge \mathbf{e}_5) \\ \mathbf{T}_z &= \frac{1}{2}(\mathbf{e}_3 \wedge \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1} \wedge \mathbf{e}_6)\end{aligned}\quad (7)$$

and 3 operators for the cross terms:

$$\begin{aligned} \mathbf{T}_{xy} &= \frac{1}{2} (\mathbf{e}_5 \wedge \mathbf{e}_1 + \mathbf{e}_4 \wedge \mathbf{e}_2) \\ \mathbf{T}_{xz} &= \frac{1}{2} (\mathbf{e}_6 \wedge \mathbf{e}_1 + \mathbf{e}_4 \wedge \mathbf{e}_3) \\ \mathbf{T}_{yz} &= \frac{1}{2} (\mathbf{e}_5 \wedge \mathbf{e}_3 + \mathbf{e}_6 \wedge \mathbf{e}_2) \end{aligned} \quad (8)$$

Then, for example,

$$\mathbf{T}_{yz} \cdot \mathbf{x} = yz \quad (9)$$

Finally a general quadratic surface is defined as the bivector \mathbf{Q}_{DCGA} with the following formula:

$$\mathbf{Q}_{DCGA} = a\mathbf{T}_{x^2} + b\mathbf{T}_{y^2} + c\mathbf{T}_{z^2} + d\mathbf{T}_{xy} + e\mathbf{T}_{yz} + f\mathbf{T}_{xz} + g\mathbf{T}_x + h\mathbf{T}_y + i\mathbf{T}_z + j\mathbf{T}_1 \quad (10)$$

Finally we check that a point \mathbf{x} is in a quadratic surface if and only if:

$$\mathbf{Q}_{DCGA} \cdot \mathbf{x} = 0 \quad (11)$$

DCGA not only supports the definition of general quadrics but also some quartic surfaces like Torus, cyclides (Dupin cyclides...).

Plane tangent to a quadratic surface

The tangent plane was defined using differential operators in DCGA. Let us consider a point \mathbf{x} whose Euclidean coordinates are (x, y, z) and a DCGA quadric \mathbf{Q}_{DCGA} defined as:

$$\mathbf{Q}_{DCGA} = a\mathbf{T}_{x^2} + b\mathbf{T}_{y^2} + c\mathbf{T}_{z^2} + d\mathbf{T}_{xy} + e\mathbf{T}_{yz} + f\mathbf{T}_{xz} + g\mathbf{T}_x + h\mathbf{T}_y + i\mathbf{T}_z + j\mathbf{T}_1 \quad (12)$$

The differential operators along the axis are defined as:

$$\begin{aligned} \mathbf{D}_x &= (\mathbf{e}_1 \wedge \mathbf{e}_{\infty 1} + \mathbf{e}_4 \wedge \mathbf{e}_{\infty 2}) \\ \mathbf{D}_y &= (\mathbf{e}_2 \wedge \mathbf{e}_{\infty 1} + \mathbf{e}_5 \wedge \mathbf{e}_{\infty 2}) \\ \mathbf{D}_z &= (\mathbf{e}_3 \wedge \mathbf{e}_{\infty 1} + \mathbf{e}_6 \wedge \mathbf{e}_{\infty 2}) \end{aligned} \quad (13)$$

Then, using the commutator product, noted as \times , the following properties hold:

$$\begin{aligned} \mathbf{D}_x \times \mathbf{q}_{DCGA} &= 2a\mathbf{T}_x + d\mathbf{T}_y + e\mathbf{T}_z + g\mathbf{T}_1 \\ \mathbf{D}_y \times \mathbf{q}_{DCGA} &= 2b\mathbf{T}_y + d\mathbf{T}_x + f\mathbf{T}_z + h\mathbf{T}_1 \\ \mathbf{D}_z \times \mathbf{q}_{DCGA} &= 2c\mathbf{T}_z + e\mathbf{T}_x + f\mathbf{T}_y + i\mathbf{T}_1 \end{aligned} \quad (14)$$

This latter formula defines the normal vector to the quadric surface at any point of the surface. It is computed using the normal vector \mathbf{n}_1 defined in the first copy CGA and \mathbf{n}_2 along the second copy of the CGA basis vectors. \mathbf{n}_1 at the considered point can be defined as follows:

$$\mathbf{n}_1 = ((\mathbf{D}_x \times \mathbf{q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_1 + ((\mathbf{D}_y \times \mathbf{q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_2 + ((\mathbf{D}_z \times \mathbf{q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_3 \quad (15)$$

Similarly, the normal vector \mathbf{n}_2 is:

$$\mathbf{n}_2 = ((\mathbf{D}_x \times \mathbf{q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_4 + ((\mathbf{D}_y \times \mathbf{q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_5 + ((\mathbf{D}_z \times \mathbf{q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_6 \quad (16)$$

Now, the definition of the plane from normal vector is:

$$\mathbf{\Pi} = (\mathbf{n}_1 + d\mathbf{e}_{\infty 1}) \wedge (\mathbf{n}_2 + d\mathbf{e}_{\infty 2}) \quad (17)$$

where d represents the orthogonal distance between the plane and the origin. Finally, the computation of the orthogonal distance can be simply performed as follows:

$$d = \mathbf{n}_1 \cdot \mathbf{x}_1 \quad (18)$$

Where \mathbf{x}_1 is the point used to form the DCGA point \mathbf{x} .

Quadric-line intersection

DCGA also supports the construction of the intersection \mathbf{p}_p of a quadric \mathbf{Q}_{DCGA} and a line \mathbf{L} . A line in DCGA can be defined as:

$$\mathbf{L} = \mathbf{l}_1 \wedge \mathbf{l}_2 \quad (19)$$

Where these two entities \mathbf{l}_1 and \mathbf{l}_2 can be expressed using the direction of the unit vector in CGA1 \mathbf{d}_1 and CGA2 \mathbf{d}_1 and a point of this line expressed in CGA1 \mathbf{x}_1 and CGA2 \mathbf{x}_2 as:

$$\mathbf{l}_1 = \mathbf{d}_1 \mathbf{I}_{\epsilon}^{-1} - (\mathbf{x}_1 \cdot (\mathbf{d}_1 \mathbf{I}_{\epsilon 1}^{-1})) \quad (20)$$

and:

$$\mathbf{l}_2 = \mathbf{d}_2 \mathbf{I}_{\epsilon}^{-1} - (\mathbf{x}_2 \cdot (\mathbf{d}_2 \mathbf{I}_{\epsilon 1}^{-1})) \quad (21)$$

Both $\mathbf{d}_1 \mathbf{I}_{\epsilon 1}$ and $\mathbf{d}_2 \mathbf{I}_{\epsilon 1}$ are 2-vectors, therefore, \mathbf{L} is a 4-vector. Note that a line can similarly be obtained by the intersection of two DCGA planes as:

$$\mathbf{L} = \mathbf{\Pi}_1 \wedge \mathbf{\Pi}_2 \quad (22)$$

Finally, the intersection is computed as:

$$\mathbf{P}_p = \mathbf{Q}_{DCGA} \wedge \mathbf{L} \quad (23)$$

Complexity of some major operations of DCGA

Let us first evaluate the computational cost of checking whether a point is on a quadric using the model of 3.1. \mathbf{Q}_{DCGA} has a total of 10 basis bivector components. For each basis bivector, at most 3 inner products (bivector \wedge bivector) are performed. Finally, the number of point component is 25. Thus, the product $\mathbf{Q}_{DCGA} \cdot \mathbf{X}$ require $25 \times 3 \times 10 = 750$ products.

The cost of the computation of the tangent plane to a quadric corresponds to the cost of, first, the normal vector \mathbf{n}_1 and second the tangent plane. Firstly, the normal vector is defined as:

$$\mathbf{n}_1 = ((\mathbf{D}_x \times \mathbf{Q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_1 + ((\mathbf{D}_y \times \mathbf{Q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_2 + ((\mathbf{D}_z \times \mathbf{Q}_{DCGA}) \cdot \mathbf{x})\mathbf{e}_3 \quad (24)$$

Equation (14) defined $\mathbf{D}_x, \mathbf{D}_y, \mathbf{D}_z$, and the commutator product of these operators with the quadric results in a 7-component bivector. Indeed, the extractions operators $\mathbf{T}_x, \mathbf{T}_y, \mathbf{T}_z$ are 2-components operator while \mathbf{T}_1 is a single component extraction operator. Each inner product with \mathbf{X} then has a computational cost of $7 \times 25 = 175$ products. This latter computation is repeated for each axis thus this results in $175 \times 3 = 525$ products.

Second, the tangent plane is obtained by using the normal vector and the orthogonal distance from the origin. The orthogonal distance is computed as:

$$d = \mathbf{n}_1 \cdot \mathbf{x}_1 \quad (25)$$

Thus, this requires the computation of 3 inner products. Finally, the tangent plane is the results of the outer product:

$$\mathbf{\Pi} = (\mathbf{n}_1 + d\mathbf{e}_{\infty 1}) \wedge (\mathbf{n}_2 + d\mathbf{e}_{\infty 2}) \quad (26)$$

Both $(\mathbf{n}_1 + d\mathbf{e}_{\infty 1})$ and $(\mathbf{n}_2 + d\mathbf{e}_{\infty 2})$ are 4-components 1-vector. Thus, the computational cost of the outer product is $4 \times 4 = 16$. Hence, the total cost of the computation of the tangent plane is $525 + 16 = 541$ products.

The cost of the computation of the intersection between a quadric and a line consists in evaluating the cost of the outer product between a DCGA line \mathbf{l} and the quadric of DCGA \mathbf{q}_{DCGA} . In the previous section, we defined a line \mathbf{l} as the 4-vector entity obtained by the outer product of the planes as follows:

$$\mathbf{L} = \mathbf{\Pi}_1 \wedge \mathbf{\Pi}_2 \quad (27)$$

A plane in DCGA is obtained as the outer product of two Conformal Geometric Algebra (CGA) planes whose number of components is 4. The result of the outer product between the two CGA planes may have non-zero components along the following components:

$$(\mathbf{e}_{14}, \mathbf{e}_{15}, \mathbf{e}_{16}, \mathbf{e}_{1\infty 2}, \mathbf{e}_{24}, \mathbf{e}_{25}, \mathbf{e}_{26}, \mathbf{e}_{2\infty 2}, \mathbf{e}_{34}, \mathbf{e}_{35}, \mathbf{e}_{36}, \mathbf{e}_{3\infty 2}, \mathbf{e}_{\infty 14}, \mathbf{e}_{\infty 15}, \mathbf{e}_{\infty 16}, \mathbf{e}_{\infty 14}, \mathbf{e}_{\infty 1\infty 2}) \quad (28)$$

Then, computing the outer product between two planes may have some results along the following basis quad-vectors:

$$(\mathbf{e}_{1245}, \mathbf{e}_{1246}, \mathbf{e}_{124\infty 2}, \mathbf{e}_{1256}, \mathbf{e}_{125\infty 2}, \mathbf{e}_{126\infty 2}, \mathbf{e}_{1345}, \mathbf{e}_{1346}, \mathbf{e}_{134\infty 2}, \mathbf{e}_{1356}, \mathbf{e}_{135\infty 2}, \mathbf{e}_{136\infty 2}, \mathbf{e}_{1\infty 145}, \mathbf{e}_{1\infty 146}, \mathbf{e}_{1\infty 14\infty 2}, \mathbf{e}_{1\infty 156}, \mathbf{e}_{1\infty 15\infty 2}, \mathbf{e}_{1\infty 16\infty 2}, \mathbf{e}_{2345}, \mathbf{e}_{2346}, \mathbf{e}_{234\infty 2}, \mathbf{e}_{2356}, \mathbf{e}_{235\infty 2}, \mathbf{e}_{236\infty 2}, \mathbf{e}_{2\infty 145}, \mathbf{e}_{2\infty 146}, \mathbf{e}_{2\infty 14\infty 2}, \mathbf{e}_{2\infty 156}, \mathbf{e}_{2\infty 15\infty 2}, \mathbf{e}_{2\infty 16\infty 2}, \mathbf{e}_{3\infty 145}, \mathbf{e}_{3\infty 146}, \mathbf{e}_{3\infty 14\infty 2}, \mathbf{e}_{3\infty 156}, \mathbf{e}_{3\infty 15\infty 2}, \mathbf{e}_{3\infty 16\infty 2}) \quad (29)$$

This latter 4-vector has thus 36 components. Then, the outer product between this quad-vector \mathbf{L} and the quadratic surface can be performed as:

$$\mathbf{Q}_{DCGA} \wedge \mathbf{L} \quad (30)$$

As the number of components of \mathbf{Q}_{DCGA} is 25 and the number of components of \mathbf{L} is 36. Then the cost of the outer product is $25 \times 36 = 900$ products. The following table summarises the computational cost of the three features computed so far for DCGA.

Table 2: Computational features in number of Geometric Algebra operations for DCGA

Feature	DCGA
point is on a quadric	725
tangent plane	541
quadric-line intersection	900

DPGA of $G_{4,4}$ DPGA was adapted from the approach of Parkin [17] in 2012 and firstly introduced in 2015 by Goldman and Mann [10] and further developed by Du and Goldman and Mann [7].

Basis and metric

DPGA $G_{4,4}$ is defined over a 8-dimensional vector space. In a similar way to DCGA, the base vectors of the space are divided into two groups: $\{\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ (corresponding to the projective Geometric Algebra vectors), and a copy of this basis $\{\mathbf{w}_0^*, \mathbf{w}_1^*, \mathbf{w}_2^*, \mathbf{w}_3^*\}$ such that $\mathbf{w}_i \mathbf{w}_i^* = 0.5 + \mathbf{w}_i \wedge \mathbf{w}_i^*, \forall i \in \{0, 1, 2, 3\}$. To have more details, we show the inner products between any basis vectors in Table 3.

Table 3: Inner product between DPGA basis vectors.

	\mathbf{w}_0	\mathbf{w}_1	\mathbf{w}_2	\mathbf{w}_3	\mathbf{w}_0^*	\mathbf{w}_1^*	\mathbf{w}_2^*	\mathbf{w}_3^*
\mathbf{w}_0	0	0	0	0	0.5	0	0	0
\mathbf{w}_1	0	0	0	0	0	0.5	0	0
\mathbf{w}_2	0	0	0	0	0	0	0.5	0
\mathbf{w}_3	0	0	0	0	0	0	0	0.5
\mathbf{w}_0^*	0.5	0	0	0	0	0	0	0
\mathbf{w}_1^*	0	0.5	0	0	0	0	0	0
\mathbf{w}_2^*	0	0	0.5	0	0	0	0	0
\mathbf{w}_3^*	0	0	0	0.5	0	0	0	0

Point of DPGA

In DPGA, the entity representing a point whose Euclidean coordinates are (x, y, z) has two definitions, namely a primal and dual. Both of the definitions are the base to construct quadrics by means of sandwiching product. The definitions

of the points are:

$$\begin{aligned}\mathbf{p} &= x\mathbf{w}_0 + y\mathbf{w}_1 + z\mathbf{w}_2 + w\mathbf{w}_3 \\ \mathbf{p}^* &= x\mathbf{w}_0^* + y\mathbf{w}_1^* + z\mathbf{w}_2^* + w\mathbf{w}_3^*\end{aligned}\quad (31)$$

Note that the dual definition denotes the fact that:

$$\mathbf{w}_i \cdot \mathbf{w}_j^* = \frac{1}{2} \delta_{ij} \quad \forall i, j = 0, \dots, 3 \quad (32)$$

Where $\delta_{ij} = 1$ if $i = j$, else 0. This corresponds to the condition of the dual stated in Section 11 of [4].

Quadrics

Again, for a general quadric defined as:

$$ax^2 + by^2 + cz^2 + dxy + eyz + fzx + gx + hy + iz + j = 0 \quad (33)$$

A quadric in DPGA is the bivector Q_{DPGA} defined as follows:

$$\begin{aligned}Q_{DPGA} &= 4a\mathbf{w}_0^* \wedge \mathbf{w}_0 + 4b\mathbf{w}_1^* \wedge \mathbf{w}_1 + 4c\mathbf{w}_2^* \wedge \mathbf{w}_2 + 4j\mathbf{w}_3^* \wedge \mathbf{w}_3 \\ &\quad + 2d(\mathbf{w}_0^* \wedge \mathbf{w}_1 + \mathbf{w}_1^* \wedge \mathbf{w}_0) + 2e(\mathbf{w}_0^* \wedge \mathbf{w}_2 + \mathbf{w}_2^* \wedge \mathbf{w}_0) \\ &\quad + 2f(\mathbf{w}_1^* \wedge \mathbf{w}_2 + \mathbf{w}_2^* \wedge \mathbf{w}_1) + 2g(\mathbf{w}_0^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_0) \\ &\quad + 2h(\mathbf{w}_1^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_1) + 2i(\mathbf{w}_2^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_2)\end{aligned}\quad (34)$$

Finally, a point (x, y, z) is in the quadric Q_{DPGA} if and only if

$$\mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}^* = 0 \quad (35)$$

Let us call $f_{DPGA} = \mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}^*$. Then to investigate numerical properties of the quadric computation, we develop the formula $\mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}^*$:

$$\begin{aligned}f_{DPGA} &= \mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}^* \\ &= \left(2ax\mathbf{w}_0 + dx\mathbf{w}_1 + ex\mathbf{w}_2 + gx\mathbf{w}_3 + 2by\mathbf{w}_1 + dy\mathbf{w}_0 + fy\mathbf{w}_2 + hy\mathbf{w}_3 \right. \\ &\quad \left. + 2cz\mathbf{w}_2 + ez\mathbf{w}_0 + fz\mathbf{w}_1 + iz\mathbf{w}_3 + 2j\mathbf{w}_3 + g\mathbf{w}_0 + h\mathbf{w}_1 + i\mathbf{w}_2 \right) \cdot \mathbf{p}^* \\ &= \left((2ax + dy + ez + g)\mathbf{w}_0 + (2by + dx + fz + h)\mathbf{w}_1 \right. \\ &\quad \left. + (2cz + ex + fy + i)\mathbf{w}_2 + (iz + gx + hy + 2j)\mathbf{w}_3 \right) \cdot \mathbf{p}^*\end{aligned}\quad (36)$$

The last inner product results in:

$$\begin{aligned}f_{DPGA} &= \mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}^* \\ &= ax^2 + 0.5dxy + 0.5exz + 0.5gx + by^2 + 0.5dxy + 0.5fyz + 0.5hy \\ &\quad + cz^2 + 0.5exz + 0.5fyz + 0.5iz + 0.5iz + 0.5gx + 0.5hy + j \\ &= ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j\end{aligned}\quad (37)$$

This latter development is the base to determine the number of operations in the computation of $\mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}^*$.

Plane tangent to a quadric

In a similar way as CGA, DPGA supports the computation of the tangent plane Π^* to a quadric \mathbf{Q}_{DPGA} at a dual point \mathbf{p}^* as follows:

$$\Pi^* = \mathbf{Q}_{DPGA} \cdot \mathbf{p}^* \quad (38)$$

Quadric-line intersection

DPGA also supports the quadric-line intersection. Given a line defined as the primal and dual $\mathbf{L} = \mathbf{x}_1 \wedge \mathbf{x}_2$ and $\mathbf{L}^* = \mathbf{x}_1^* \wedge \mathbf{x}_2^*$ and the quadratic surface bivector \mathbf{Q}_{DPGA} . The intersection bivector called \mathbf{P}_p is defined as follows:

$$\mathbf{P}_p = (\mathbf{L}^* \wedge \mathbf{Q}_{DPGA} \wedge \mathbf{L}) \cdot \mathbf{I} \quad (39)$$

where \mathbf{I} is the pseudo-scalar of $G_{4,4}$ defined as:

$$\mathbf{I} = \mathbf{w}_0 \wedge \mathbf{w}_1 \wedge \mathbf{w}_2 \wedge \mathbf{w}_3 \wedge \mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \quad (40)$$

Complexity of some major operations of DPGA

\mathbf{Q}_{DPGA} has a total of 16 basis bivector components. For each basis bivector, 2 inner products are performed. Thus, the first product $\mathbf{P} \cdot \mathbf{Q}_{DCGA}$ will require $4 \times 2 \times 16 = 128$ inner products. As previously seen, the resulting entity is a vector with 4 components. Hence, the second inner product requires $4 \times 4 = 16$ products. This results in 144 products.

Let us now evaluate the cost of the intersection between a quadratic surface \mathbf{Q}_{DPGA} and a line \mathbf{L} and \mathbf{L}^* . The line \mathbf{L}^* is obtained by the outer product of two points \mathbf{x}_1 and \mathbf{x}_2 whose number of components is 4. Thus a line \mathbf{L} has 6 components along the bivector basis:

$$(\mathbf{w}_0 \wedge \mathbf{w}_1, \mathbf{w}_0 \wedge \mathbf{w}_2, \mathbf{w}_0 \wedge \mathbf{w}_3, \mathbf{w}_1 \wedge \mathbf{w}_2, \mathbf{w}_1 \wedge \mathbf{w}_3, \mathbf{w}_2 \wedge \mathbf{w}_3) \quad (41)$$

and a line \mathbf{L}^* has the following bivector basis components:

$$(\mathbf{w}_0^* \wedge \mathbf{w}_1^*, \mathbf{w}_0^* \wedge \mathbf{w}_2^*, \mathbf{w}_0^* \wedge \mathbf{w}_3^*, \mathbf{w}_1^* \wedge \mathbf{w}_2^*, \mathbf{w}_1^* \wedge \mathbf{w}_3^*, \mathbf{w}_2^* \wedge \mathbf{w}_3^*) \quad (42)$$

The number of components of the quadratic surface is 16 and the number of components of the line is 6. Then, the computational cost of the outer product $\mathbf{L}^* \wedge \mathbf{Q}_{DPGA}$ is $6 \times 16 = 96$ outer products. The result is a 4-vector and may have components along the quad-vector basis:

$$\begin{aligned} &(\mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_0, \mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_1, \mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_2, \\ &\mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3, \mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_0, \mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_1, \\ &\mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_2, \mathbf{w}_0^* \wedge \mathbf{w}_1^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_3, \mathbf{w}_0^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_0, \\ &\mathbf{w}_0^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_1, \mathbf{w}_0^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_2, \mathbf{w}_0^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_3, \\ &\mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_0, \mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_1, \mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_2, \\ &\mathbf{w}_1^* \wedge \mathbf{w}_2^* \wedge \mathbf{w}_3^* \wedge \mathbf{w}_3) \end{aligned} \quad (43)$$

Thus, the resulting entity has 16 components. Furthermore, the line \mathbf{l} has 6 components. Hence, the cost of the final outer product is $16 \times 6 = 96$ outer products. Finally, the total operation cost is thus $96 + 96 = 192$ products.

The cost of the computation of the tangent plane at a point \mathbf{p} is the cost of the following product:

$$\pi^* = \mathbf{Q}_{DPGA} \cdot \mathbf{p}^* \quad (44)$$

Considering the fact that the number of components of \mathbf{p}^* is 4 and the number of components of \mathbf{Q}_{DPGA} is 16. Then the computational cost of the computation of the tangent plane is $16 \times 4 = 64$ products.

QCGA of $\mathbb{G}_{9,6}$ Let us evaluate the computational cost of checking whether a point is on a quadratic surface. \mathbf{Q}^* has a total of 12 basis vector components. For each basis vector, at most 1 inner product is performed, please refer to the Equation (??). Finally, the number of point component is 12. Thus, the product $\mathbf{x} \cdot \mathbf{Q}^*$ requires at most $12 \times 12 = 144$ products.

The computation of the tangent plane is performed by first the computation of the normal vector as:

$$\begin{aligned} \mathbf{n}_\epsilon = & \left(((\mathbf{x} \cdot \mathbf{e}_1)\mathbf{e}_{\infty 1} + (\mathbf{x} \cdot \mathbf{e}_2)\mathbf{e}_{\infty 4} + (\mathbf{x} \cdot \mathbf{e}_3)\mathbf{e}_{\infty 5} + \mathbf{e}_1) \cdot \mathbf{q}^* \right) \mathbf{e}_1 + \\ & \left(((\mathbf{x} \cdot \mathbf{e}_2)\mathbf{e}_{\infty 2} + (\mathbf{x} \cdot \mathbf{e}_1)\mathbf{e}_{\infty 4} + (\mathbf{x} \cdot \mathbf{e}_3)\mathbf{e}_{\infty 6} + \mathbf{e}_2) \cdot \mathbf{q}^* \right) \mathbf{e}_2 + \\ & \left(((\mathbf{x} \cdot \mathbf{e}_3)\mathbf{e}_{\infty 3} + (\mathbf{x} \cdot \mathbf{e}_1)\mathbf{e}_{\infty 5} + (\mathbf{x} \cdot \mathbf{e}_2)\mathbf{e}_{\infty 6} + \mathbf{e}_3) \cdot \mathbf{q}^* \right) \mathbf{e}_3. \end{aligned} \quad (45)$$

This computation required the inner product between a vector with 12 components and another vector with 4 components. This computation is repeated for each Euclidean basis vector thus the computation of the normal vector requires $3 \times 4 \times 12 = 144$ inner products.

Then, the tangent plane is computed using the normal vector as follows:

$$\pi^* = \mathbf{n}_\epsilon + \frac{1}{3}(\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3}) \sqrt{-2(\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3}) \cdot \mathbf{x}}. \quad (46)$$

This computation requires the computation of an inner product of a vector with 3 components $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ with a 12 component-vector. This means $12 \times 3 = 36$ products. Thus the total number of inner products required in the computation of the tangent plane is $144 + 36 = 180$ products.

The final computational feature is the quadric-line intersection. In QCGA, this simply consists in computing the outer product:

$$\mathbf{c}^* = \mathbf{q}^* \wedge \mathbf{l}^* \quad (47)$$

The number of components of \mathbf{q}^* is 12 as already seen. In QCGA, we defined a line with the 6 Plücker coefficients as:

$$\mathbf{l}^* = 3\mathbf{m}\mathbf{I}_\epsilon + (\mathbf{e}_{\infty 3} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1}) \wedge \mathbf{n}\mathbf{I}_\epsilon. \quad (48)$$

The number of components of both \mathbf{m} and \mathbf{n} is 3. The outer product $(\mathbf{e}_{\infty 3} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1}) \wedge \mathbf{n}\mathbf{I}_\epsilon$ yields a copy of the 3 components of \mathbf{n} along $\mathbf{e}_{\infty 1}, \mathbf{e}_{\infty 2}, \mathbf{e}_{\infty 3}$ basis vectors. Thus, the number of components of \mathbf{l}^* is $3 \times 3 + 3 = 12$. Finally, the cost of the outer product between \mathbf{q}^* and \mathbf{l}^* is $12 \times 12 = 144$ products.

The table 4 summarizes the computational features of the proposed framework compared to DPGA and DCGA. We remark that the computation of the tan-

Table 4: Comparison of the computational features between QCGA, DCGA, DPGA in number of Geometric Algebra operations

Feature	DPGA	DCGA	QCGA
point is on a quadric	144	750	144
tangent plane	64	541	180
quadric-line intersection	192	300	144

gent plane is more efficient using DPGA whereas the intersection between a quadratic surface and a line requires less computations using QCGA. Furthermore, some versors are not defined in some models.

4 Mapping

As a practical application, it might be interesting to construct a quadratic surface from 9 points then rotating this quadratic surface. For the moment, QCGA is the only approach, in Geometric Algebra, that can construct quadratic surface from 9 points but it does not yet support all the transformations. Furthermore, the last chapter shows that some operations are worth doing in a certain framework. These points are our motivation for defining new operators that convert quadratic surfaces between the three presented frameworks.

The key idea is that for any entities representing quadric surface in QCGA, DCGA and DPGA, it is possible to convert the entity such that all the coefficients of the quadrics:

$$ax^2 + by^2 + cz^2 + dxy + eyz + fzx + gx + hy + iz + j = 0, \quad (49)$$

can be extracted easily.

5 DCGA reciprocal operators

This means defining reciprocal operators for DCGA:

$$\begin{aligned} \mathbf{T}^{x^2} &= \mathbf{e}_1 \wedge \mathbf{e}_4 & \mathbf{T}_{y^2} &= \mathbf{e}_2 \wedge \mathbf{e}_5 \\ \mathbf{T}^{z^2} &= \mathbf{e}_3 \wedge \mathbf{e}_6 & \mathbf{T}_1 &= \mathbf{e}_{o1} \wedge \mathbf{e}_{o2} \end{aligned} \quad (50)$$

along with the 6 following:

$$\begin{aligned} \mathbf{T}^x &= (\mathbf{e}_1 \wedge \mathbf{e}_{o2} + \mathbf{e}_{o1} \wedge \mathbf{e}_4) \\ \mathbf{T}^y &= (\mathbf{e}_2 \wedge \mathbf{e}_{o2} + \mathbf{e}_{o1} \wedge \mathbf{e}_5) \\ \mathbf{T}^z &= (\mathbf{e}_3 \wedge \mathbf{e}_{o2} + \mathbf{e}_{o1} \wedge \mathbf{e}_6) \\ \mathbf{T}^{xy} &= (\mathbf{e}_1 \wedge \mathbf{e}_5 + \mathbf{e}_2 \wedge \mathbf{e}_4) \\ \mathbf{T}^{xz} &= (\mathbf{e}_1 \wedge \mathbf{e}_6 + \mathbf{e}_3 \wedge \mathbf{e}_4) \\ \mathbf{T}^{yz} &= (\mathbf{e}_3 \wedge \mathbf{e}_5 + \mathbf{e}_2 \wedge \mathbf{e}_6) \end{aligned} \quad (51)$$

These reciprocal operators verify the following properties:

$$\begin{aligned} \mathbf{T}^{x^2} \cdot \mathbf{T}_{x^2} &= 1, \mathbf{T}^{y^2} \cdot \mathbf{T}_{y^2} = 1, \mathbf{T}^{z^2} \cdot \mathbf{T}_{z^2} = 1, \mathbf{T}^{xy} \cdot \mathbf{T}_{xy} = 1, \mathbf{T}^{xz} \cdot \mathbf{T}_{xz} = 1, \\ \mathbf{T}^{yz} \cdot \mathbf{T}_{yz} &= 1, \mathbf{T}^x \cdot \mathbf{T}_x = 1, \mathbf{T}^y \cdot \mathbf{T}_y = 1, \mathbf{T}^z \cdot \mathbf{T}_z = 1, \mathbf{T}^1 \cdot \mathbf{T}_1 = 1 \end{aligned} \quad (52)$$

Then, given \mathbf{q}_{DCGA} the entity representing a quadratic surface of DCGA, any coefficients of this quadratic surface (49) can be extracted as:

$$\begin{aligned} \mathbf{T}^{x^2} \cdot \mathbf{q}_{DCGA} &= a, \mathbf{T}^{y^2} \cdot \mathbf{q}_{DCGA} = b, \mathbf{T}^{z^2} \cdot \mathbf{q}_{DCGA} = c, \mathbf{T}^{xy} \cdot \mathbf{q}_{DCGA} = d, \\ \mathbf{T}^{xz} \cdot \mathbf{q}_{DCGA} &= e, \mathbf{T}^{yz} \cdot \mathbf{q}_{DCGA} = f, \mathbf{T}^x \cdot \mathbf{q}_{DCGA} = g, \mathbf{T}^y \cdot \mathbf{q}_{DCGA} = h, \\ \mathbf{T}^z \cdot \mathbf{q}_{DCGA} &= i, \mathbf{T}^1 \cdot \mathbf{q}_{DCGA} = j \end{aligned} \quad (53)$$

The coefficients of the implicit form of the surface can then be extracted as:

$$\begin{aligned} a &= \mathbf{T}^{x^2} \cdot \mathbf{q}_{DCGA} & b &= \mathbf{T}^{y^2} \cdot \mathbf{q}_{DCGA} & c &= \mathbf{T}^{z^2} \cdot \mathbf{q}_{DCGA} \\ d &= \mathbf{T}^{xy} \cdot \mathbf{q}_{DCGA} & e &= \mathbf{T}^{xz} \cdot \mathbf{q}_{DCGA} & f &= \mathbf{T}^{yz} \cdot \mathbf{q}_{DCGA} \\ g &= \mathbf{T}^x \cdot \mathbf{q}_{DCGA} & h &= \mathbf{T}^y \cdot \mathbf{q}_{DCGA} & i &= \mathbf{T}^z \cdot \mathbf{q}_{DCGA} \\ j &= \mathbf{T}^1 \cdot \mathbf{q}_{DCGA} \end{aligned} \quad (54)$$

The construction of a DCGA point was already explained in the previous section and defined in [8]. The reciprocal operation requires the computation of the normalization point $\hat{\mathbf{x}}$ of DCGA that we define as:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\mathbf{x} \cdot (\mathbf{e}_{\infty 1} \wedge \mathbf{e}_{\infty 2})} \quad (55)$$

The extraction of the Euclidean components (x, y, z) of a normalized point $\hat{\mathbf{x}}$ of DCGA can be performed as follows:

$$x = \hat{\mathbf{x}} \cdot \mathbf{e}_1, \quad y = \hat{\mathbf{x}} \cdot \mathbf{e}_2, \quad z = \hat{\mathbf{x}} \cdot \mathbf{e}_3 \quad (56)$$

6 DPGA reciprocal operators

In a similar way, let us note \mathbf{W} reciprocal operators for DPGA

$$\begin{aligned} \mathbf{W}^{x^2} &= \mathbf{w}_0^* \wedge \mathbf{w}_0 & \mathbf{W}^{y^2} &= \mathbf{w}_1^* \wedge \mathbf{w}_1 \\ \mathbf{W}^{z^2} &= \mathbf{w}_2^* \wedge \mathbf{w}_2 & \mathbf{W}^{xy} &= 2\mathbf{w}_1^* \wedge \mathbf{w}_0 \\ \mathbf{W}^{xz} &= 2\mathbf{w}_2^* \wedge \mathbf{w}_0 & \mathbf{W}^{yz} &= 2\mathbf{w}_2^* \wedge \mathbf{w}_1 \\ \mathbf{W}^x &= 2\mathbf{w}_3^* \wedge \mathbf{w}_0 & \mathbf{W}^y &= 2\mathbf{w}_3^* \wedge \mathbf{w}_1 \\ \mathbf{W}^z &= 2\mathbf{w}_3^* \wedge \mathbf{w}_2 & \mathbf{W}^1 &= \mathbf{W}_3^* \wedge \mathbf{w}_3 \end{aligned} \quad (57)$$

Again, the following properties hold:

$$\begin{aligned} \mathbf{W}^{x^2} \cdot \mathbf{W}_{x^2} &= 1, \mathbf{W}^{y^2} \cdot \mathbf{W}_{y^2} = 1, \mathbf{W}^{z^2} \cdot \mathbf{W}_{z^2} = 1, \mathbf{W}^{xy} \cdot \mathbf{W}_{xy} = 1, \mathbf{W}^{xz} \cdot \mathbf{W}_{xz} = 1, \\ \mathbf{W}^{yz} \cdot \mathbf{W}_{yz} &= 1, \mathbf{W}^x \cdot \mathbf{W}_x = 1, \mathbf{W}^y \cdot \mathbf{W}_y = 1, \mathbf{W}^z \cdot \mathbf{W}_z = 1, \mathbf{W}^1 \cdot \mathbf{W}_1 = 1 \end{aligned} \quad (58)$$

Then, given \mathbf{q}_{DPGA} the entity representing a quadratic of DPGA, any coefficients of this quadratic surface (49) can be extracted as:

$$\begin{aligned} \mathbf{W}^{x^2} \cdot \mathbf{q}_{DPGA} &= a, \mathbf{W}^{y^2} \cdot \mathbf{q}_{DPGA} = b, \mathbf{W}^{z^2} \cdot \mathbf{q}_{DPGA} = c, \mathbf{W}^{xy} \cdot \mathbf{q}_{DPGA} = d, \\ \mathbf{W}^{xz} \cdot \mathbf{q}_{DPGA} &= e, \mathbf{W}^{yz} \cdot \mathbf{q}_{DPGA} = f, \mathbf{W}^x \cdot \mathbf{q}_{DPGA} = g, \mathbf{W}^y \cdot \mathbf{q}_{DPGA} = h, \\ \mathbf{W}^z \cdot \mathbf{q}_{DPGA} &= i, \mathbf{W}^1 \cdot \mathbf{q}_{DPGA} = j \end{aligned} \quad (59)$$

In a similar way as in projective geometry, the construction of a finite point of DPGA requires to add a homogeneous component 1 to the Euclidean components. The normalization of a point merely consists in dividing all the components by its \mathbf{w}_3 components (or \mathbf{w}_3^* for the dual form) if it is a non-zero component.

7 QCGA reciprocal operators

For QCGA, quadratic surfaces can be either representing using the primal form or the dual form. We define the reciprocal operators for the dual form. Indeed, if one consider the primal form, then this would consist in computing the dual of the primal and apply the following reciprocal operators:

$$\begin{aligned} \mathbf{Q}^{x^2} &= \frac{1}{2}\mathbf{e}_{\infty 1} & \mathbf{Q}^{y^2} &= \frac{1}{2}\mathbf{e}_{\infty 2} \\ \mathbf{Q}^{z^2} &= \frac{1}{2}\mathbf{e}_{\infty 3} & \mathbf{Q}^{xy} &= \mathbf{e}_{\infty 4} \\ \mathbf{Q}^{xz} &= \mathbf{e}_{\infty 5} & \mathbf{Q}^{yz} &= \mathbf{e}_{\infty 6} \\ \mathbf{Q}^x &= \mathbf{e}_1 & \mathbf{Q}^y &= \mathbf{e}_2 \\ \mathbf{Q}^z &= \mathbf{e}_3 & \mathbf{Q}^1 &= \mathbf{e}_{o1} + \mathbf{e}_{o2} + \mathbf{e}_{o3} \end{aligned} \quad (60)$$

Given a general quadratic \mathbf{q}^* whose coefficients are (a, b, c, \dots, j) , the properties of these operators are as follows:

$$\begin{aligned} \mathbf{Q}^{x^2} \cdot \mathbf{q}^* &= a, \mathbf{Q}^{y^2} \cdot \mathbf{q}^* = b, \mathbf{Q}^{z^2} \cdot \mathbf{q}^* = c, \mathbf{Q}^{xy} \cdot \mathbf{q}^* = d, \mathbf{Q}^{xz} \cdot \mathbf{q}^* = e, \\ \mathbf{Q}^{yz} \cdot \mathbf{q}^* &= f, \mathbf{Q}^x \cdot \mathbf{q}^* = g, \mathbf{Q}^y \cdot \mathbf{q}^* = h, \mathbf{Q}^z \cdot \mathbf{q}^* = i, \mathbf{Q}^1 \cdot \mathbf{q}^* = j \end{aligned} \quad (61)$$

The construction of a QCGA point was already explained in the previous section. The reciprocal operation requires the computation of the normalization point $\hat{\mathbf{x}}$ of QCGA.

Proposition 71 *For QCGA point \mathbf{x} , the normalization is merely computed through an averaging of $\mathbf{e}_{o1}, \mathbf{e}_{o2}, \mathbf{e}_{o3}$ components thus of \mathbf{e}_o component, namely as:*

$$-\frac{\mathbf{x}}{\mathbf{x} \cdot \mathbf{e}_\infty} \quad (62)$$

Proof. A scale α on \mathbf{x} acts the same way on all null basis vectors of \mathbf{x} :

$$\alpha \mathbf{x} = \alpha \mathbf{x}_e + \frac{1}{2} \alpha (x^2 \mathbf{e}_{\infty 1} + y^2 \mathbf{e}_{\infty 2} + z^2 \mathbf{e}_{\infty 3}) + xy \alpha \mathbf{e}_{\infty 4} + xz \alpha \mathbf{e}_{\infty 5} + yz \alpha \mathbf{e}_{\infty 6} + \alpha \mathbf{e}_{o1} + \alpha \mathbf{e}_{o2} + \alpha \mathbf{e}_{o3} \quad (63)$$

The metric of QCGA indicates (see Table ??):

$$\begin{aligned} \alpha \mathbf{x} \cdot \mathbf{e}_{\infty 1} &= -\alpha \\ \alpha \mathbf{x} \cdot \mathbf{e}_{\infty 2} &= -\alpha \\ \alpha \mathbf{x} \cdot \mathbf{e}_{\infty 3} &= -\alpha \end{aligned} \quad (64)$$

Thus, if $\alpha \neq 0$:

$$\begin{aligned} \frac{-3\alpha \mathbf{x}}{\alpha \mathbf{x} \cdot (\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3})} \cdot \mathbf{e}_{\infty 1} &= -\frac{-3\alpha \mathbf{x}}{-3\alpha} \cdot \mathbf{e}_{\infty 1} \\ &= \mathbf{x} \cdot \mathbf{e}_{\infty 1} = -1 \end{aligned} \quad (65)$$

A similar result is obtained with $\mathbf{e}_{\infty 2}$ and $\mathbf{e}_{\infty 3}$:

$$\frac{-3\alpha \mathbf{x}}{\alpha \mathbf{x} \cdot (\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3})} \cdot \mathbf{e}_{\infty 2} = \mathbf{x} \cdot \mathbf{e}_{\infty 2} = -1 \quad (66)$$

$$\frac{-3\alpha \mathbf{x}}{\alpha \mathbf{x} \cdot (\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3})} \cdot \mathbf{e}_{\infty 3} = \mathbf{x} \cdot \mathbf{e}_{\infty 3} = -1 \quad (67)$$

Thus, we checked that for any scaled points $\mathbf{x}_1, \mathbf{x}_2$:

$$\frac{\mathbf{x}_1}{\mathbf{x}_1 \cdot \mathbf{e}_\infty} \cdot \frac{\mathbf{x}_2}{\mathbf{x}_2 \cdot \mathbf{e}_\infty} = -\frac{1}{2} \|\mathbf{x}_{1e} - \mathbf{x}_{2e}\|^2 \quad (68)$$

The extraction of the Euclidean components (x, y, z) of a normalized point $\hat{\mathbf{x}}$ of DCGA can be performed as follows:

$$x = \hat{\mathbf{x}} \cdot \mathbf{e}_1, \quad y = \hat{\mathbf{x}} \cdot \mathbf{e}_2, \quad z = \hat{\mathbf{x}} \cdot \mathbf{e}_3 \quad (69)$$

8 Test

We tested this approach by defining an ellipsoid from 9 points using QCGA. Then we rotate it using DPGA and back-convert the rotated ellipsoid into QCGA

framework. In terms of Geometric Algebra computations, first we compute the quadratic:

$$\mathbf{q}^* = (\mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \cdots \wedge \mathbf{x}_9 \wedge \mathbf{I}_o^{\triangleright})^* \quad (70)$$

Then, we apply the extraction operators of QCGA to convert the QCGA quadratic to DPGA quadratic.

$$\begin{aligned} \mathbf{q}_{DPGA} = & 4(\mathbf{Q}^{x^2} \cdot \mathbf{q}^*) \mathbf{w}_0^* \wedge \mathbf{w}_0 + 4(\mathbf{Q}^{y^2} \cdot \mathbf{q}^*) \mathbf{w}_1^* \wedge \mathbf{w}_1 + 4(\mathbf{Q}^{z^2} \cdot \mathbf{q}^*) \mathbf{w}_2^* \wedge \mathbf{w}_2 \\ & + 4(\mathbf{Q}^1 \cdot \mathbf{q}^*) \mathbf{w}_3^* \wedge \mathbf{w}_3 \\ & + 2(\mathbf{Q}^{xy} \cdot \mathbf{q}^*) (\mathbf{w}_0^* \wedge \mathbf{w}_1 + \mathbf{w}_1^* \wedge \mathbf{w}_0) \\ & + 2(\mathbf{Q}^{xz} \cdot \mathbf{q}^*) (\mathbf{w}_0^* \wedge \mathbf{w}_2 + \mathbf{w}_2^* \wedge \mathbf{w}_0) \\ & + 2(\mathbf{Q}^{yz} \cdot \mathbf{q}^*) (\mathbf{w}_1^* \wedge \mathbf{w}_2 + \mathbf{w}_2^* \wedge \mathbf{w}_1) \\ & + 2(\mathbf{Q}^x \cdot \mathbf{q}^*) (\mathbf{w}_0^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_0) \\ & + 2(\mathbf{Q}^y \cdot \mathbf{q}^*) (\mathbf{w}_1^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_1) \\ & + 2(\mathbf{Q}^z \cdot \mathbf{q}^*) (\mathbf{w}_2^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_2) \end{aligned} \quad (71)$$

The rotation is now performed as follows:

$$\mathbf{q}_{DPGA} = \mathbf{R} \mathbf{q}_{DPGA} \mathbf{R}^{-1} \quad (72)$$

The rotor \mathbf{R} is defined as:

$$\mathbf{R} = \exp\left(\frac{1}{2} \theta \mathbf{w}_i \mathbf{w}_j^*\right) \quad (73)$$

where $i \neq j$

The final step is to back-convert the resulting quadric to the QCGA framework. It is merely computed using the QCGA extraction operators as follows:

$$\begin{aligned} \mathbf{q}^* = & - (2(\mathbf{W}^{x^2} \cdot \mathbf{q}_{DPGA}) \mathbf{e}_{01} + 2(\mathbf{W}^{y^2} \cdot \mathbf{q}_{DPGA}) \mathbf{e}_{02} + 2(\mathbf{W}^{z^2} \cdot \mathbf{q}_{DPGA}) \mathbf{e}_{03} \\ & + (\mathbf{W}^{xy} \cdot \mathbf{q}_{DPGA}) \mathbf{e}_{04} + (\mathbf{W}^{xz} \cdot \mathbf{q}_{DPGA}) \mathbf{e}_{05} + (\mathbf{W}^{yz} \cdot \mathbf{q}_{DPGA}) \mathbf{e}_{06}) \\ & + ((\mathbf{W}^x \cdot \mathbf{q}_{DPGA}) \mathbf{e}_1 + (\mathbf{W}^y \cdot \mathbf{q}_{DPGA}) \mathbf{e}_2 + (\mathbf{W}^z \cdot \mathbf{q}_{DPGA}) \mathbf{e}_3) \\ & - \frac{(\mathbf{W}^1 \cdot \mathbf{q}_{DPGA})}{3} (\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3}). \end{aligned} \quad (74)$$

Note that the program can be found in the plugin folder of the git repository previously shown.

9 Conclusion

In this paper, we focused on a hybrid approach to deal with quadratic surfaces. After presenting the main approaches to represent and manipulate quadratic surfaces, we introduced a new hybrid Geometric Algebra approach. This approach unifies all the models of Geometric Algebra into one more general approach that allows to represent any quadratic surface either using control points or from the coefficients of its implicit form. We showed that the proposed method also enables to easily extract geometric properties, like the curvature, in an efficient way. For the following, we seek for a generalisation of this approach for the representation of quadratic and cubic surfaces. Some frameworks are considered and all of these models require high dimensional frameworks.

References

1. BREUILS, S., NOZICK, V., AND FUCHS, L. Garamon: Geometric algebra library generator.
2. BREUILS, S., NOZICK, V., SUGIMOTO, A., AND HITZER, E. Quadric conformal geometric algebra of $\mathbb{R}^{9,6}$. *Advances in Applied Clifford Algebras* 28, 2 (Mar 2018), 35.
3. BUCHHOLZ, S., TACHIBANA, K., AND HITZER, E. M. Optimal learning rates for clifford neurons. In *International conference on artificial neural networks* (2007), Springer, pp. 864–873.
4. DORAN, C., HESTENES, D., SOMMEN, F., AND VAN ACKER, N. Lie groups as spin groups. *Journal of Mathematical Physics* 34, 8 (1993), 3642–3669.
5. DORST, L., FONTIJNE, D., AND MANN, S. *Geometric Algebra for Computer Science, An Object-Oriented Approach to Geometry*. Morgan Kaufmann, 2007.
6. DORST, L., AND VAN DEN BOOMGAARD, R. An analytical theory of mathematical morphology. In *Mathematical Morphology and its Applications to Signal Processing* (1993), pp. 245–250.
7. DU, J., GOLDMAN, R., AND MANN, S. Modeling 3D Geometry in the Clifford Algebra $\mathbb{R}^{4,4}$. *Advances in Applied Clifford Algebras* 27, 4 (Dec 2017), 3039–3062.
8. EASTER, ROBERT BENJAMIN AND HITZER, ECKHARD. Double conformal geometric algebra. *Advances in Applied Clifford Algebras* 27, 3 (2017), 2175–2199.
9. GLASSNER, A. S. *An introduction to ray tracing*. Elsevier, 1989.
10. GOLDMAN, R., AND MANN, S. $\mathbb{R}(4, 4)$ as a computational framework for 3-dimensional computer graphics. *Advances in Applied Clifford Algebras* 25, 1 (Mar 2015), 113–149.
11. GREGORY, A. L., LASENBY, J., AND AGARWAL, A. The elastic theory of shells using geometric algebra. *Royal Society open science* 4, 3 (2017), 170065.
12. HESTENES, D. The zitterbewegung interpretation of quantum mechanics. *Foundations of Physics* 20, 10 (Oct 1990), 1213–1232.
13. HESTENES, D. *New foundations for classical mechanics*, vol. 15. Springer Science & Business Media, 2012.
14. HITZER, E. Geometric operations implemented by conformal geometric algebra neural nodes. *arXiv preprint arXiv:1306.1358* (2013).
15. LUO, W., HU, Y., YU, Z., YUAN, L., AND LÜ, G. A Hierarchical Representation and Computation Scheme of Arbitrary-dimensional Geometrical Primitives Based on CGA. *Advances in Applied Clifford Algebras* 27, 3 (Sep 2017), 1977–1995.
16. PAPAETHYMIU, M., AND PAPAGIANNAKIS, G. Real-time rendering under distant illumination with conformal geometric algebra. *Mathematical Methods in the Applied Sciences* (2017).
17. PARKIN, S. T. A model for quadric surfaces using geometric algebra. *Unpublished*, October (2012).
18. PERWASS, C. *Geometric algebra with applications in engineering*, vol. 4 of *Geometry and Computing*. Springer, 2009.
19. VINCE, J. *Geometric algebra for computer graphics*. Springer Science & Business Media, 2008.
20. ZAMORA-ESQUIVEL, J. G 6,3 geometric algebra; description and implementation. *Advances in Applied Clifford Algebras* 24, 2 (Jun 2014), 493–514.
21. ZHU, S., YUAN, S., LI, D., LUO, W., YUAN, L., AND YU, Z. Mvtree for hierarchical network representation based on geometric algebra subspace. *Advances in Applied Clifford Algebras* 28, 2 (Apr 2018), 39.