HAL
open science

Transverse approach to geometric algebra models for manipulating quadratic surfaces

Stéphane Breuils, Vincent Nozick, Laurent Fuchs, Akihiro Sugimoto

To cite this version:

Stéphane Breuils, Vincent Nozick, Laurent Fuchs, Akihiro Sugimoto. Transverse approach to geometric algebra models for manipulating quadratic surfaces. Computer Graphics International (CGI2019), Jun 2019, Calgary, Canada. hal-02050767v3

HAL Id: hal-02050767

https://hal.science/hal-02050767v3

Submitted on 1 Apr 2019

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.

Transverse approach to geometric algebra models for manipulating quadratic surfaces

Stéphane Breuils[1], Vincent Nozick[1], Laurent Fuchs[2] and Akihiro Sugimoto[3]

[1] Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI,
UMR 8049, Université Paris-Est Marne-la-Vallée, France
{stephane.breuils, vincent.nozick}@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 in the geometric algebra community and some frameworks to represent, transform, and intersect these quadratic surfaces have been proposed. To the best of our knowledge, however, no framework has yet proposed that supports all the operations required to completely handle these surfaces. Some existing frameworks do not allow the construction of quadratic surfaces from control points while some do not allow to transform these quadratic surfaces. Although a framework does not exit that covers all the required operations, if we consider all already proposed frameworks together, then all the operations over quadratic surfaces are covered there. This paper presents an approach that transversely uses different frameworks. We employ a framework to represent any quadratic surfaces either using control points or the coefficients of its implicit form and then map the representation into another framework so that we can transform them and compute their intersection. Our approach also allows us 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 [11] as well as in classical mechanics [12]. Geometric algebra has also found some interesting applications in geographic data manipulations [15, 20]. 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. For image data analysis, on the other hand, we can find the usefulness of geometric algebra in mathematical morphology [5] and in neural networking [2, 13]. In the geometric algebra community, quadratic surfaces gain more and more attention, and some frameworks to represent, transform, and intersect these quadratic surfaces have been proposed.

There exist three main approaches to deal with quadratic surfaces in geometric algebra. The first one, introduced in [8], is called double conformal geometric algebra (DCGA), $\mathbb{G}_{8,2}$. It is capable of representing quadratic surfaces from the coefficients of their implicit form. The second one is double projective geometric algebra (DPGA), $\mathbb{G}_{4,4}$, whose definition was firstly introduced in [10] and has been further developed in [7]. This approach is based on a duplication of $\mathbb{R}^4$ and it represents quadratic surfaces from the coefficients of their implicit form, as bivectors. However, it cannot construct quadratic surfaces from control points. The third one was introduced in [1] and is denoted as quadric conformal geometric algebra (QCGA), $\mathbb{G}_{9,6}$. QCGA allows to define any general quadratic surface from 9 control points, and to represent objects by only 1 or 2-vectors. QCGA is capable of constructing quadratic surfaces either using control points or implicit equations as 1-vector. QCGA also allows to efficiently intersect quadratic surfaces. However, it does not yet allow all geometric transformations over quadratic surfaces. In order to enhance the usefulness of geometric algebra for the geometry and the computer graphics communities, a new framework that allows to represent and manipulate quadratic surfaces has to be developed. It is the main purpose of this paper.

1.1 Contributions

We propose a new approach that transversely uses the three above mentioned geometric algebra models to compensate drawbacks of each model and show that it is possible to not only represent quadratic surfaces using either control points or implicit coefficients but also transform these quadratic surfaces using versors. More precisely, we employ a model that allows us to represent quadratic surfaces, and convert them into another model that allows us to transform them using versors, and then convert the result back into the original model. With our approach, the tangent planes to a quadratic surfaces and intersection of quadratic surfaces can be computed.

1.2 Notations

Following the state-of-the-art usage in [4] and [18], upper-case bold letters denote blades (blade $\mathbf{A}$) 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 vector space dimension is denoted by 2^d, where d is the number of basis blades $\mathbf{e}_i$ of grade 1.

2 Geometric algebra models for quadratic surfaces

2.1 Measure for evaluating complexity of models

In order to compare the operations for different frameworks, we need a criterion to evaluate models. This requires the ability of determining the complexity of operations in a model. Our complexity estimation is made through two simple 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 then assume that an upper bound to the number of required products is at most uv products of scalars, as shown in the definition of the outer product [14].

The equation of the inner product is our base for the second assumption. Furthermore, we need to use the 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. The inner product between 1-vectors and 2-vectors, on the other hand, requires two inner products for each pair of components of the two multivectors, that is to say $2uv$ products. These models will be used for determining the complexity of each operation. Let us now explain in more details these models.

2.2 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 operations required in computer graphics. The purpose here is to propose an approach that transversely uses geometric algebra models to efficiently handle quadratic surfaces. The considered operations over the surfaces are:

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

One of the applications requiring the above operations is the precise visualization using ray-tracer [9]. There exist three main geometric algebra frameworks to manipulate general quadratic surfaces: DCGA of $\mathbb{G}_{8,2}$ [8], DPGA of $\mathbb{G}_{4,4}$ [17, 7], and QGCA of $\mathbb{G}_{9,6}$ [1].

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. In more details, DCGA of $\mathbb{G}_{8,2}$ is defined over a 10-dimensional vector space. The base vectors

of the space are basically divided into two groups: $\{\mathbf{e}_{o1}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_{\infty 1}\}$, corresponding to the CGA vectors, and a copy of this basis $\{\mathbf{e}_{o2}, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_{\infty 2}\}$. 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).

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

$$\mathrm{a}x^2 + \mathrm{b}y^2 + \mathrm{c}z^2 + \mathrm{d}xy + \mathrm{e}yz + \mathrm{f}zx + \mathrm{g}x + \mathrm{h}y + \mathrm{i}z + \mathrm{j} = 0. \tag{1}$$

In DCGA, 10 extraction operators $\{\mathbf{T}_{x^2}, \mathbf{T}_{y^2}, \mathbf{T}_{z^2}, \mathbf{T}_{xy}, \mathbf{T}_{xz}, \mathbf{T}_{yz}, \mathbf{T}_x, \mathbf{T}_y, \mathbf{T}_z, \mathbf{T}_1\}$ are defined (see [8]) such that the inner product of these operators and a point results in Equation 1. DCGA not only supports the definition of general quadratic surfaces but also some quartic surfaces like Torus, cyclides (Dupin cyclides, etc.).

Complexity of some major operations: Let us first evaluate the computational cost of checking whether a point is on a quadratic surface using the model of 2.1. $\mathbf{Q}_{\mathrm{DCGA}}$ has 10 basis bivector components in total. For each basis bivector, at most 3 inner products (bivector $\wedge$ bivector) are required. The number of point components is 25. Thus, the product $\mathbf{Q}_{\mathrm{DCGA}} \cdot \mathbf{X}$ requires $25 \times 3 \times 10 = 750$ products.

This paragraph details the cost of the computation of the tangent plane to a quadratic surface, defined in [8] as:

$$\boldsymbol{\Pi} = (\mathbf{n}_1 + \mathrm{d}\mathbf{e}_{\infty 1}) \wedge (\mathbf{n}_2 + \mathrm{d}\mathbf{e}_{\infty 2}). \tag{2}$$

The normal vector is defined as the commutator product of some differential operators and the quadratic surface resulting in a 7-component bivector. Each inner product with $\mathbf{X}$ then has the cost of $7 \times 25 = 175$ products. This latter computation is repeated for each axis; thus, this results in $175 \times 3 = 525$ products. The computation of the distance d, on the other hand, consists of merely 3 inner products. Both operands in this equation 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 third operation is the intersection between a quadratic surface and a line. This computation is, unfortunately, not defined in DCGA.

DPGA of $\mathbb{G}_{4,4}$.

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]. DPGA is defined over a 8-dimensional vector space. Similarly 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 $\mathbf{w}_i \mathbf{w}_i^* = 0.5 + \mathbf{w}_i \wedge \mathbf{w}_i^*$, $\forall i \in \{0, 1, 2, 3\}$. In DPGA, the entity representing a point whose Euclidean coordinates are (x, y, z) has two definitions, namely, primal and dual. Both definitions are the base to construct quadratic surfaces by means of the sandwiching

Table 1: Formulas of DPGA involved in the main computations for computer graphics

Feature	DPGA
point is on a quadric	$\mathbf{p} \cdot Q_{\text{DPGA}} \cdot \mathbf{p}^*$
tangent plane	$Q_{\text{DPGA}} \cdot \mathbf{p}^*$
quadric-line intersection	$(\mathbf{L}^* \wedge Q_{\text{DPGA}} \wedge \mathbf{L}) \cdot \mathbf{I}$

product. The definitions of the points are:

$$\mathbf{p} = x\mathbf{w}_0 + y\mathbf{w}_1 + z\mathbf{w}_2 + w\mathbf{w}_3, \qquad \mathbf{p}^* = x\mathbf{w}_0^* + y\mathbf{w}_1^* + z\mathbf{w}_2^* + w\mathbf{w}_3^*. \tag{3}$$

Note that the dual definition denotes the fact that

$$\mathbf{w}_i \cdot \mathbf{w}_j^* = \frac{1}{2}\delta_{i,j} \; (\forall i, j = 0, \cdots 3), \tag{4}$$

where $\delta_{i,j} = 1$ if $i = j$, 0 otherwise. This corresponds to the condition of the dual stated in Section 11 of [3].

Quadratic surfaces: A quadratic surface in DPGA is the bivector Q_{DPGA} defined as follows:

$$\begin{aligned} Q_{\text{DPGA}} = & 4\mathrm{a}\mathbf{w}_0^* \wedge \mathbf{w}_0 + 4\mathrm{b}\mathbf{w}_1^* \wedge \mathbf{w}_1 + 4\mathrm{c}\mathbf{w}_2^* \wedge \mathbf{w}_2 + 4\mathrm{j}\mathbf{w}_3^* \wedge \mathbf{w}_3 \\ & + 2\mathrm{d}(\mathbf{w}_0^* \wedge \mathbf{w}_1 + \mathbf{w}_1^* \wedge \mathbf{w}_0) + 2\mathrm{e}(\mathbf{w}_0^* \wedge \mathbf{w}_2 + \mathbf{w}_2^* \wedge \mathbf{w}_0) \\ & + 2\mathrm{f}(\mathbf{w}_1^* \wedge \mathbf{w}_2 + \mathbf{w}_2^* \wedge \mathbf{w}_1) + 2\mathrm{g}(\mathbf{w}_0^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_0) \\ & + 2\mathrm{h}(\mathbf{w}_1^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_1) + 2\mathrm{i}(\mathbf{w}_2^* \wedge \mathbf{w}_3 + \mathbf{w}_3^* \wedge \mathbf{w}_2). \end{aligned} \tag{5}$$

A point (x, y, z) is in the quadratic surface Q_{DPGA} if and only if

$$\mathbf{p} \cdot Q_{\text{DPGA}} \cdot \mathbf{p}^* = 0. \tag{6}$$

Table 1 summarises the computations involved in three main operations used for computer graphics.

Complexity of some major operations: Q_{DPGA} has a total of 16 basis bivector components. For each basis bivector, 2 inner products are required. Thus, the first product $\mathbf{P} \cdot Q_{\text{DPGA}}$ requires $4 \times 2 \times 16 = 128$ 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 in total.

Let us now evaluate the cost of the intersection between a quadratic surface 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. 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 $\mathbf{L}^* \wedge Q_{\text{DPGA}}$ is $6 \times 16 = 96$ outer products. The result is a 4-vector and 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. The total operation cost is thus $96 + 96 = 192$ products.

Considering the fact that the number of components of $\mathbf{p}^*$ is 4 and the number of components of Q_{DPGA} is 16, 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. The definition of a quadratic surface in QCGA is a 1-vector, called $\mathbf{Q}^*$, which has a total of 12 basis vector components. Furthermore, 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 firstly computing the normal vector. This computation requires the inner product between a vector with 12 components and another vector with 4 components. This 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:

$$\boldsymbol{\pi}^* = \mathbf{n}_\epsilon + \frac{1}{3}(\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3})\sqrt{-2(\mathbf{e}_{o1} + \mathbf{e}_{o2} + \mathbf{e}_{o3}) \cdot \mathbf{x}}. \tag{7}$$

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 for computing the tangent plane is $144 + 36 = 180$ products.

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

$$\mathbf{C}^* = \mathbf{Q}^* \wedge \mathbf{L}^* \tag{8}$$

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

$$\mathbf{L}^* = 3\,\mathbf{m}\,\mathbf{I}_\epsilon + (\mathbf{e}_{\infty 3} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 1}) \wedge \mathbf{n}\,\mathbf{I}_\epsilon. \tag{9}$$

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 $3 \times 3 + 3 = 12$. The cost of the outer product between $\mathbf{Q}^*$ and $\mathbf{L}^*$ is thus $12 \times 12 = 144$ products.

Table 2 summarises the complexity of DPGA, DCGA, and QCGA for computing features. We remark that the computation of the tangent plane is more efficient if we use DPGA whereas the intersection between a quadratic surface and a line requires less computations if we use QCGA. Furthermore, geometric transformations are not yet defined in QCGA.

3 Mapping between the three models for quadratic surfaces

As one of practical applications, we consider constructing a quadratic surface from 9 points then rotating this quadratic surface. To the best of our knowledge,

Table 2: Numbers of operations required for computation in DPGA, DCGA, and QCGA.

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

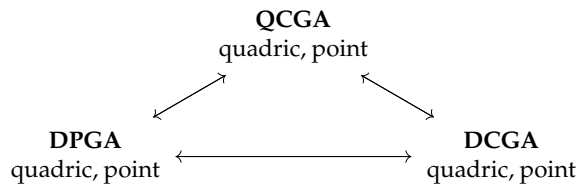

Fig. 1: Encapsulation of the three models of points and quadrics.

QCGA is the only approach that can construct a quadratic surface from 9 points. But QCGA does not yet support all the transformations. Furthermore, the last section shows that it is more computationally efficient to perform the quadric-line intersection in the QCGA model whereas the computation of the tangent plane or the normal vector at a point of quadratic surface is more efficient in the DPGA model. Moreover, if we represent both a Dupin cyclide and a quadratic surface in a same way as [6], then we need DCGA. The above observation is our motivation for defining new operators that convert quadratic surfaces between the three models, see Fig. 1.

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

$$\mathrm{a}x^2 + \mathrm{b}y^2 + \mathrm{c}z^2 + \mathrm{d}xy + \mathrm{e}yz + \mathrm{f}zx + \mathrm{g}x + \mathrm{h}y + \mathrm{i}z + \mathrm{j} = 0 \tag{10}$$

can be extracted easily.

3.1 DCGA reciprocal operators

This means defining reciprocal operators for DCGA:

$$\mathbf{T}^{x^2} = \mathbf{e}_1 \wedge \mathbf{e}_4, \quad \mathbf{T}^{y^2} = \mathbf{e}_2 \wedge \mathbf{e}_5, \quad \mathbf{T}^{z^2} = \mathbf{e}_3 \wedge \mathbf{e}_6, \quad \mathbf{T}^{1} = \mathbf{e}_{o1} \wedge \mathbf{e}_{o2}, \tag{11}$$

along with the 6 following:

$$\begin{aligned} &\mathbf{T}^{x} = \Big(\mathbf{e}_1 \wedge \mathbf{e}_{o2} + \mathbf{e}_{o1} \wedge \mathbf{e}_4\Big),\ \mathbf{T}^{y} = \Big(\mathbf{e}_2 \wedge \mathbf{e}_{o2} + \mathbf{e}_{o1} \wedge \mathbf{e}_5\Big), \\ &\mathbf{T}^{z} = \Big(\mathbf{e}_3 \wedge \mathbf{e}_{o2} + \mathbf{e}_{o1} \wedge \mathbf{e}_6\Big),\ \mathbf{T}^{xy} = \Big(\mathbf{e}_1 \wedge \mathbf{e}_5 + \mathbf{e}_2 \wedge \mathbf{e}_4\Big), \\ &\mathbf{T}^{xz} = \Big(\mathbf{e}_1 \wedge \mathbf{e}_6 + \mathbf{e}_3 \wedge \mathbf{e}_4\Big),\ \ \mathbf{T}^{yz} = \Big(\mathbf{e}_3 \wedge \mathbf{e}_5 + \mathbf{e}_2 \wedge \mathbf{e}_6\Big). \end{aligned} \tag{12}$$

Given the DCGA extraction operators presented in Section 2.2, our defined reciprocal operators verify the following properties:

$$\begin{aligned}&\mathbf{T}^{x^2}\cdot\mathbf{T}_{x^2}=1,\ \mathbf{T}^{y^2}\cdot\mathbf{T}_{y^2}=1,\ \mathbf{T}^{z^2}\cdot\mathbf{T}_{z^2}=1,\ \mathbf{T}^{xy}\cdot\mathbf{T}_{xy}=1,\ \mathbf{T}^{xz}\cdot\mathbf{T}_{xz}=1,\\&\mathbf{T}^{yz}\cdot\mathbf{T}_{yz}=1,\ \mathbf{T}^{x}\cdot\mathbf{T}_{x}=1,\ \mathbf{T}^{y}\cdot\mathbf{T}_{y}=1,\ \mathbf{T}^{z}\cdot\mathbf{T}_{z}=1,\ \mathbf{T}^{1}\cdot\mathbf{T}_{1}=1.\end{aligned}\tag{13}$$

Then, given $\mathbf{Q}_{\mathrm{DCGA}}$, the entity representing a quadratic surface of DCGA, any coefficients of this quadratic surface (10) can be extracted as:

$$\begin{aligned}&\mathbf{T}^{x^2}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{a},\ \mathbf{T}^{y^2}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{b},\ \mathbf{T}^{z^2}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{c},\ \mathbf{T}^{xy}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{d},\\&\mathbf{T}^{xz}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{e},\mathbf{T}^{yz}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{f},\ \mathbf{T}^{x}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{g},\ \mathbf{T}^{y}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{h},\\&\mathbf{T}^{z}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{i},\ \mathbf{T}^{1}\cdot\mathbf{Q}_{\mathrm{DCGA}}=\mathrm{j}.\end{aligned}\tag{14}$$

The construction of a DCGA point is explained in Section 2.2 and defined in [8]. The reciprocal operation requires the computation of the normalization point $\hat{\mathbf{x}}$ of DCGA that we define as:

$$\hat{\mathbf{X}}=-\frac{\mathbf{X}}{\mathbf{X}\cdot(\mathbf{e}_{\infty 1}\wedge\mathbf{e}_{\infty 2})}.\tag{15}$$

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

$$x=\hat{\mathbf{X}}\cdot(\mathbf{e}_1\wedge\mathbf{e}_{\infty 2}),\qquad y=\hat{\mathbf{X}}\cdot(\mathbf{e}_2\wedge\mathbf{e}_{\infty 2}),\qquad z=\hat{\mathbf{X}}\cdot(\mathbf{e}_3\wedge\mathbf{e}_{\infty 2}).\tag{16}$$

3.2 DPGA reciprocal operators

Let us denote by $\mathbf{W}$ reciprocal operators for DPGA:

$$\begin{aligned}&\mathbf{W}^{x^2}=\mathbf{w}_0^*\wedge\mathbf{w}_0,\ \ \mathbf{W}^{y^2}=\mathbf{w}_1^*\wedge\mathbf{w}_1,\ \ \mathbf{W}^{z^2}=\mathbf{w}_2^*\wedge\mathbf{w}_2,\ \ \mathbf{W}^{xy}=2\mathbf{w}_1^*\wedge\mathbf{w}_0,\\&\mathbf{W}^{xz}=2\mathbf{w}_2^*\wedge\mathbf{w}_0,\ \mathbf{W}^{yz}=2\mathbf{w}_2^*\wedge\mathbf{w}_1,\ \mathbf{W}^{x}=2\mathbf{w}_3^*\wedge\mathbf{w}_0,\ \mathbf{W}^{y}=2\mathbf{w}_3^*\wedge\mathbf{w}_1,\\&\mathbf{W}^{z}=2\mathbf{w}_3^*\wedge\mathbf{w}_2,\ \mathbf{W}^{1}=\mathbf{w}_3^*\wedge\mathbf{w}_3.\end{aligned}\tag{17}$$

Given Q_{DPGA}, the entity representing a quadratic of DPGA, any coefficients of this quadratic surface (10) can be extracted as:

$$\begin{aligned}&\mathbf{W}^{x^2}\cdot Q_{\mathrm{DPGA}}=\mathrm{a},\ \mathbf{W}^{y^2}\cdot Q_{\mathrm{DPGA}}=\mathrm{b},\ \mathbf{W}^{z^2}\cdot Q_{\mathrm{DPGA}}=\mathrm{c},\ \mathbf{W}^{xy}\cdot Q_{\mathrm{DPGA}}=\mathrm{d},\\&\mathbf{W}^{xz}\cdot Q_{\mathrm{DPGA}}=\mathrm{e},\ \mathbf{W}^{yz}\cdot Q_{\mathrm{DPGA}}=\mathrm{f},\ \mathbf{W}^{x}\cdot Q_{\mathrm{DPGA}}=\mathrm{g},\ \mathbf{W}^{y}\cdot Q_{\mathrm{DPGA}}=\mathrm{h},\\&\mathbf{W}^{z}\cdot Q_{\mathrm{DPGA}}=\mathrm{i},\ \mathbf{W}^{1}\cdot Q_{\mathrm{DPGA}}=\mathrm{j}.\end{aligned}\tag{18}$$

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 of 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.

3.3 QCGA reciprocal operators

For QCGA, quadratic surfaces can be represented using either the primal form or the dual form. We define the reciprocal operators for the dual form. When considering the primal form, we have only to compute the dual of the primal and then apply the following reciprocal operators:

$$\begin{aligned}
&\mathbf{Q}^{x^2} = \frac{1}{2}\mathbf{e}_{o1}, \quad \mathbf{Q}^{y^2} = \frac{1}{2}\mathbf{e}_{o2}, \quad \mathbf{Q}^{z^2} = \frac{1}{2}\mathbf{e}_{o3}, \quad \mathbf{Q}^{xy} = \mathbf{e}_{o4},\\
&\mathbf{Q}^{xz} = \mathbf{e}_{o5}, \qquad \mathbf{Q}^{yz} = \mathbf{e}_{o6}, \qquad \mathbf{Q}^{x} = \mathbf{e}_{1}, \qquad \mathbf{Q}^{y} = \mathbf{e}_{2},\\
&\mathbf{Q}^{z} = \mathbf{e}_{3}, \qquad \mathbf{Q}^{1} = \mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3}.
\end{aligned} \tag{19}$$

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

$$\begin{aligned}
&\mathbf{Q}^{x^2}\cdot\mathbf{Q}^* = \mathrm{a},\ \mathbf{Q}^{y^2}\cdot\mathbf{Q}^* = \mathrm{b},\ \mathbf{Q}^{z^2}\cdot\mathbf{Q}^* = \mathrm{c},\ \mathbf{Q}^{xy}\cdot\mathbf{Q}^* = \mathrm{d},\ \mathbf{Q}^{xz}\cdot\mathbf{Q}^* = \mathrm{e},\\
&\mathbf{Q}^{yz}\cdot\mathbf{Q}^* = \mathrm{f},\ \mathbf{Q}^{x}\cdot\mathbf{Q}^* = \mathrm{g},\ \mathbf{Q}^{y}\cdot\mathbf{Q}^* = \mathrm{h},\ \mathbf{Q}^{z}\cdot\mathbf{Q}^* = \mathrm{i},\ \mathbf{Q}^{1}\cdot\mathbf{Q}^* = \mathrm{j}.
\end{aligned} \tag{20}$$

The construction of a QCGA point is explained in [1]. However, the reciprocal operation requires the computation of the normalization point $\hat{\mathbf{x}}$ of QCGA which is missing in [1].

Proposition 31 *For a 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 as:*

$$-\frac{\mathbf{x}}{\mathbf{x}\cdot\mathbf{e}_\infty}. \tag{21}$$

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

$$\begin{aligned}
\alpha\mathbf{x} = \alpha\mathbf{x}_\epsilon + \tfrac{1}{2}\alpha(x^2\mathbf{e}_{\infty 1} + y^2\mathrm{w}\mathbf{e}_{\infty 2} + z^2\mathbf{e}_{\infty 3}) + xy\alpha\mathbf{e}_{\infty 4} + xz\alpha\mathbf{e}_{\infty 5} + yz\alpha\mathbf{e}_{\infty 6}\\
+\alpha\mathbf{e}_{o1} + \alpha\mathbf{e}_{o2} + \alpha\mathbf{e}_{o3}.
\end{aligned} \tag{22}$$

The metric of QCGA indicates (see [1]):

$$\alpha\mathbf{x}\cdot\mathbf{e}_{\infty 1} = -\alpha, \quad \alpha\mathbf{x}\cdot\mathbf{e}_{\infty 2} = -\alpha, \quad \alpha\mathbf{x}\cdot\mathbf{e}_{\infty 3} = -\alpha. \tag{23}$$

Thus, if $\alpha \neq 0$:

$$\begin{aligned}
\frac{-3\alpha\mathbf{x}}{\alpha\mathbf{x}\cdot(\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3})}\cdot\mathbf{e}_{\infty 1} &= -\frac{-3\alpha\mathbf{x}}{-3\alpha}\cdot\mathbf{e}_{\infty 1}\\
&= \mathbf{x}\cdot\mathbf{e}_{\infty 1} = -1.
\end{aligned} \tag{24}$$

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

$$\frac{-3\alpha\mathbf{x}}{\alpha\mathbf{x}\cdot(\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3})}\cdot\mathbf{e}_{\infty 2} = \mathbf{x}\cdot\mathbf{e}_{\infty 2} = -1. \tag{25}$$

$$\frac{-3\alpha\mathbf{x}}{\alpha\mathbf{x}\cdot(\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3})}\cdot\mathbf{e}_{\infty 3} = \mathbf{x}\cdot\mathbf{e}_{\infty 3} = -1. \tag{26}$$

Table 3: Geometric operations allowed in either QCGA, DPGA or DCGA, where ✓means possible and ✗ means not.

Opération	DPGA	DCGA	QCGA
quadric from control points	✗	✗	✓
point ∈ quadric	✓	✓	✓
tangent plane	✓	✓	✓
quadric-line intersection	✓	✗	✓
quadric-quadric intersection	✗	✗	✓
transformations	✓	✓	✗
Quartic surfaces	✗	✓	✗

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

$$\frac{\mathbf{x}_1}{\mathbf{x}_1 \cdot \mathbf{e}_\infty} \cdot \frac{\mathbf{x}_2}{\mathbf{x}_2 \cdot \mathbf{e}_\infty} = -\frac{1}{2} \left\| \mathbf{x}_{1\epsilon} - \mathbf{x}_{2\epsilon} \right\|^2. \tag{27}$$

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

$$x = \hat{\mathbf{x}} \cdot \mathbf{e}_1, \quad y = \hat{\mathbf{x}} \cdot \mathbf{e}_2, \quad z = \hat{\mathbf{x}} \cdot \mathbf{e}_3. \tag{28}$$

3.4 How to choose the right model?

Given a geometric operation and this general framework, a question arises that which model we should choose among QCGA, DPGA and DCGA. To answer this question, we merely consider two criteria, namely, (1) if the operation is defined and (2) on which model it is the most computationally efficient. This is illustrated in Table 2 and Table 3.

3.5 Example

We tested our approach by defining an ellipsoid from 9 points using QCGA. Then we rotate it using DPGA and back-convert the rotated ellipsoid into the QCGA framework. In terms of geometric algebra computations, first we compute the quadric:

$$\mathbf{Q}^* = (\mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \cdots \wedge \mathbf{x}_9 \wedge \mathbf{I}_o^{\triangleright})^*. \tag{29}$$

Then, we apply the extraction operators of QCGA to convert the QCGA quadric to its corresponding DPGA quadric.

$$\begin{aligned} Q_{\text{DPGA}} =& 4(\mathbf{Q}^{x^2}\cdot\mathbf{Q}^*)\mathbf{w}_0^*\wedge\mathbf{w}_0 + 4(\mathbf{Q}^{y^2}\cdot\mathbf{Q}^*)\mathbf{w}_1^*\wedge\mathbf{w}_1 + 4(\mathbf{Q}^{z^2}\cdot\mathbf{Q}^*)\mathbf{w}_2^*\wedge\mathbf{w}_2 \\ &+4(\mathbf{Q}^{1}\cdot\mathbf{Q}^*)\mathbf{w}_3^*\wedge\mathbf{w}_3 + 2(\mathbf{Q}^{xy}\cdot\mathbf{Q}^*)(\mathbf{w}_0^*\wedge\mathbf{w}_1 + \mathbf{w}_1^*\wedge\mathbf{w}_0) \\ &+2(\mathbf{Q}^{xz}\cdot\mathbf{Q}^*)(\mathbf{w}_0^*\wedge\mathbf{w}_2 + \mathbf{w}_2^*\wedge\mathbf{w}_0) + 2(\mathbf{Q}^{yz}\cdot\mathbf{Q}^*)(\mathbf{w}_1^*\wedge\mathbf{w}_2 + \mathbf{w}_2^*\wedge\mathbf{w}_1) \\ &+2(\mathbf{Q}^{x}\cdot\mathbf{Q}^*)(\mathbf{w}_0^*\wedge\mathbf{w}_3 + \mathbf{w}_3^*\wedge\mathbf{w}_0) + 2(\mathbf{Q}^{y}\cdot\mathbf{Q}^*)(\mathbf{w}_1^*\wedge\mathbf{w}_3 + \mathbf{w}_3^*\wedge\mathbf{w}_1) \\ &+2(\mathbf{Q}^{z}\cdot\mathbf{Q}^*)(\mathbf{w}_2^*\wedge\mathbf{w}_3 + \mathbf{w}_3^*\wedge\mathbf{w}_2). \end{aligned} \tag{30}$$

The rotation is now performed as follows:

$$Q_{\text{DPGA}} = \mathbf{R}Q_{\text{DPGA}}\mathbf{R}^{-1}. \tag{31}$$

The rotor $\mathbf{R}$ is defined as:

$$\mathbf{R} = \exp(\frac{1}{2}\theta\mathbf{w}_i\mathbf{w}_j^*), \tag{32}$$

where $i \neq j$.

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

$$\begin{aligned} \mathbf{Q}^* = &-\big(2(\mathbf{W}^{x^2}\cdot Q_{\text{DPGA}})\mathbf{e}_{o1} + 2(\mathbf{W}^{y^2}\cdot Q_{\text{DPGA}})\mathbf{e}_{o2} + 2(\mathbf{W}^{z^2}\cdot Q_{\text{DPGA}})\mathbf{e}_{o3} \\ &+ (\mathbf{W}^{xy}\cdot Q_{\text{DPGA}})\mathbf{e}_{o4} + (\mathbf{W}^{xz}\cdot Q_{\text{DPGA}})\mathbf{e}_{o5} + (\mathbf{W}^{yz}\cdot Q_{\text{DPGA}})\mathbf{e}_{o6}\big) \\ &+ \big((\mathbf{W}^{x}\cdot Q_{\text{DPGA}})\mathbf{e}_1 + (\mathbf{W}^{y}\cdot Q_{\text{DPGA}})\mathbf{e}_2 + (\mathbf{W}^{z}\cdot Q_{\text{DPGA}})\mathbf{e}_3\big) \\ &- \frac{(\mathbf{W}^{1}\cdot Q_{\text{DPGA}})}{3}(\mathbf{e}_{\infty 1} + \mathbf{e}_{\infty 2} + \mathbf{e}_{\infty 3}). \end{aligned} \tag{33}$$

Note that the program can be found in the plugin folder of the git repository https://git.renater.fr/garamon.git.

4 Conclusion

In this paper, we focused on an approach to deal with quadratic surfaces. After presenting the main geometric algebras to represent and manipulate quadratic surfaces, we introduced an approach that transversely uses the main geometric algebras. 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. For the following, we seek for a generalization 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., Sugimoto, A., and Hitzer, E. Quadric conformal geometric algebra of $\mathbb{R}^{9,6}$. *Advances in Applied Clifford Algebras 28*, 2 (Mar 2018), 35.

2. 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.
3. DORAN, C., HESTENES, D., SOMMEN, F., AND VAN ACKER, N. Lie groups as spin groups. *Journal of Mathematical Physics 34*, 8 (1993), 3642–3669.
4. DORST, L., FONTIJNE, D., AND MANN, S. *Geometric Algebra for Computer Science, An Object-Oriented Approach to Geometry*. Morgan Kaufmann, 2007.
5. 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.
6. DRUOTON, L., FUCHS, L., GARNIER, L., AND LANGEVIN, R. The non-degenerate dupin cyclides in the space of spheres using geometric algebra. *Advances in Applied Clifford Algebras 24*, 2 (2014), 515–532.
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. 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. *New foundations for classical mechanics*, vol. 15. Springer Science & Business Media, 2012.
13. HITZER, E. Geometric operations implemented by conformal geometric algebra neural nodes. *arXiv preprint arXiv:1306.1358* (2013).
14. LEOPARDI, P. A generalized FFT for Clifford algebras. *Bulletin of Belgian Mathematical Society 11* (2004), 663–688.
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. PAPAEFTHYMIOU, 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. 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.