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-02050767v2>

HAL Id: hal-02050767
https://hal.archives-ouvertes.fr/hal-02050767v2
Submitted on 4 Mar 2019 (v2), last revised 1 Apr 2019 (v3)
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
² XLIM-ASALI, UMR 7252, Université de Poitiers, Poitiers, France
³ National Institute of Informatics, Tokyo 101-8430, Japan,

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 quadratics 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 online.

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 \( G_{6,3} \). In this model, an axis-aligned quadratic surface \( Q \) can be defined as:

\[
Q = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_6
\]

(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 \( G_{8,2} \) defined by Easter and Hitzer \[8\]. Second, a framework of \( G_{4,4} \) 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 \( G_{9,6} \).

**DCGA of \( G_{8,2} \)**

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 \( G_{8,2} \) is defined over a 10-dimensional vector space. The base vectors of the space are basically divided into two groups: \( \{ e_{01}, e_1, e_2, e_3, e_{\infty 1} \} \), corresponding to the CGA vectors defined in Chapter 1, and a copy of this basis \( \{ e_{02}, e_4, e_5, e_6, e_{\infty 2} \} \). 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>
<thead>
<tr>
<th></th>
<th>( e_{01} )</th>
<th>( e_1 )</th>
<th>( e_2 )</th>
<th>( e_3 )</th>
<th>( e_{\infty 1} )</th>
<th>( e_{02} )</th>
<th>( e_4 )</th>
<th>( e_5 )</th>
<th>( e_6 )</th>
<th>( e_{\infty 2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_{01} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_1 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_2 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_{\infty 1} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_{02} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_4 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_5 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_6 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( e_{\infty 2} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**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 \(x_1\) and \(x_2\) as:

\[
\begin{align*}
  x_1 &= e_{o1} + xe_{1} + ye_{2} + ze_{3} + \frac{1}{2}(x^2 + y^2 + z^2)e_{o01} \\
  x_2 &= e_{o2} + xe_{4} + ye_{5} + ze_{6} + \frac{1}{2}(x^2 + y^2 + z^2)e_{o02}
\end{align*}
\]

(2)

The embedding of a DCGA point is defined as follows:

\[
X = x_1 \wedge x_2
\]

(3)

The development of this operation results in:

\[
X = x^2e_1 \wedge e_4 + y^2e_2 \wedge e_5 + z^2e_3 \wedge e_6 + xyz(e_1 \wedge e_5 + e_2 \wedge e_4) + xz(e_1 \wedge e_6 + e_3 \wedge e_4) + yz(e_2 \wedge e_6 + e_3 \wedge e_5) + xe_1 \wedge e_4 + e_1 \wedge e_5 + e_2 \wedge e_4) + ye_1 \wedge e_5 + e_3 \wedge e_5) + ze_1 \wedge e_6 + e_3 \wedge e_6) + \frac{1}{2}(x^2 + y^2 + z^2)(e_{o1} \wedge e_{o02} + e_{o01} \wedge e_{o2}) + \frac{1}{2}(x^2 + y^2 + z^2)(e_{o1} \wedge e_{o3} + e_{o01} \wedge e_{o1}) + \frac{1}{2}(x^2 + y^2 + z^2)(e_{o2} + e_{o02} + e_{o01} \wedge e_{o5}) + \frac{1}{2}(x^2 + y^2 + z^2)(e_{o3} + e_{o03} + e_{o01} \wedge e_{o6}) + \frac{1}{2}(x^2 + y^2 + z^2)^2(e_{o1} \wedge e_{o02} + e_{o01} \wedge e_{o2})
\]

(4)

This high number of components is due to the fact that the representation of a point of DCGA was designed to not 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 \(x\). For a general quadric defined as:

\[
a x^2 + by^2 + cz^2 + dxy + eyz + fzx + gx + hy + iz + j = 0
\]

(5)

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

\[
\begin{align*}
  T_{x^2} &= e_4 \wedge e_1 \\
  T_{y^2} &= e_5 \wedge e_2 \\
  T_{z^2} &= e_6 \wedge e_3 \\
  T_1 &= -e_{o01} \wedge e_{o02}
\end{align*}
\]

(6)

along with the 3 operators for the linear part:

\[
\begin{align*}
  T_x &= \frac{1}{2}\left(e_1 \wedge e_{o02} + e_{o01} \wedge e_4\right) \\
  T_y &= \frac{1}{2}\left(e_2 \wedge e_{o02} + e_{o01} \wedge e_5\right) \\
  T_z &= \frac{1}{2}\left(e_3 \wedge e_{o02} + e_{o01} \wedge e_6\right)
\end{align*}
\]

(7)
and 3 operators for the cross terms:

\[
T_{xy} = \frac{1}{2} \left( e_5 \wedge e_1 + e_4 \wedge e_2 \right)
\]

\[
T_{xz} = \frac{1}{2} \left( e_6 \wedge e_1 + e_4 \wedge e_3 \right)
\]

\[
T_{yz} = \frac{1}{2} \left( e_5 \wedge e_3 + e_6 \wedge e_2 \right)
\]

(8)

Then, for example,

\[
T_{yz} \cdot x = yz
\]

(9)

Finally a general quadratic surface is defined as the bivector \( Q_{DCGA} \) with the following formula:

\[
Q_{DCGA} = aT_{x^2} + bT_{y^2} + cT_{z^2} +dT_{xy} + eT_{yz} + fT_{xz} + gT_x + hT_y + iT_z + jT_1
\]

(10)

Finally we check that a point \( x \) is in a quadratic surface if and only if:

\[
Q_{DCGA} \cdot x = 0
\]

(11)

DCGA not only supports the definition of general quadrics but also some quadric 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 \( x \) whose Euclidean coordinates are \((x, y, z)\) and a DCGA quadric \( Q_{DCGA} \) defined as:

\[
Q_{DCGA} = aT_{x^2} + bT_{y^2} + cT_{z^2} +dT_{xy} + eT_{yz} + fT_{xz} + gT_x + hT_y + iT_z + jT_1
\]

(12)

The differential operators along the axis are defined as:

\[
D_x = (e_1 \wedge e_{01} + e_4 \wedge e_{02})
\]

\[
D_y = (e_2 \wedge e_{01} + e_5 \wedge e_{02})
\]

\[
D_z = (e_3 \wedge e_{01} + e_6 \wedge e_{02})
\]

(13)

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

\[
D_x \times q_{DCGA} = 2aT_x +dT_y + eT_z + gT_1
\]

\[
D_y \times q_{DCGA} = 2bT_y +dT_x + fT_z + hT_1
\]

\[
D_z \times q_{DCGA} = 2cT_z + eT_x + fT_y + iT_1
\]

(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 \( n_1 \) defined in the first copy CGA and \( n_2 \) along the second copy of the CGA basis vectors. \( n_1 \) at the considered point can be defined as follows:

\[
n_1 = ((D_x \times q_{DCGA}) \cdot x)e_1 + ((D_y \times q_{DCGA}) \cdot x)e_2 + ((D_z \times q_{DCGA}) \cdot x)e_3
\]

(15)
Similarly, the normal vector \( \mathbf{n}_2 \) is:

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

(16)

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

\[
\Pi = (\mathbf{n}_1 + d\mathbf{e}_\infty) \wedge (\mathbf{n}_2 + d\mathbf{e}_\infty)
\]  

(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:

\[
d = \mathbf{n}_1 \cdot \mathbf{x}_1
\]  

(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 \( \mathbf{Q}_{\text{DCGA}} \) and a line \( \mathbf{L} \). A line in DCGA can be defined as:

\[
\mathbf{L} = \mathbf{I}_1 \wedge \mathbf{I}_2
\]  

(19)

Where these two entities \( \mathbf{I}_1 \) and \( \mathbf{I}_2 \) 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:

\[
\mathbf{I}_1 = \mathbf{d}_1 \mathbf{I}_e^{-1} - (\mathbf{x}_1 \cdot (\mathbf{d}_1 \mathbf{I}_e^{-1}))
\]  

(20)

and:

\[
\mathbf{I}_2 = \mathbf{d}_2 \mathbf{I}_e^{-1} - (\mathbf{x}_2 \cdot (\mathbf{d}_2 \mathbf{I}_e^{-1}))
\]  

(21)

Both \( \mathbf{d}_1 \mathbf{I}_e \) and \( \mathbf{d}_2 \mathbf{I}_e \) 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:

\[
\mathbf{L} = \Pi_1 \wedge \Pi_2
\]  

(22)

Finally, the intersection is computed as:

\[
\mathbf{p}_p = \mathbf{Q}_{\text{DCGA}} \wedge \mathbf{L}
\]  

(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. \( \mathbf{Q}_{\text{DCGA}} \) 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 \( \mathbf{Q}_{\text{DCGA}} \cdot \mathbf{X} \) 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:

\[
\mathbf{n}_1 = ((\mathbf{D}_x \times \mathbf{Q}_{\text{DCGA}}) \cdot \mathbf{x})\mathbf{e}_1 + ((\mathbf{D}_y \times \mathbf{Q}_{\text{DCGA}}) \cdot \mathbf{x})\mathbf{e}_2 + ((\mathbf{D}_z \times \mathbf{Q}_{\text{DCGA}}) \cdot \mathbf{x})\mathbf{e}_3
\]  

(24)
Equation (14) defined $D_x, D_y, D_z,$ and the commutator product of these operators with the quadric results in a 7-component bivector. Indeed, the extractions operators $T_x, T_y, T_z$ are 2-components operator while $T_1$ is a single component extraction operator. Each inner product with $X$ 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:

$$d = n_1 \cdot x_1$$

(25)

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

$$\Pi = (n_1 + de_{o1}) \wedge (n_2 + de_{o2})$$

(26)

Both $(n_1 + de_{o1})$ and $(n_2 + de_{o2})$ 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 $l$ and the quadric of DCGA $q_{DCGA}$. In the previous section, we defined a line $l$ as the 4-vector entity obtained by the outer product of the planes as follows:

$$L = \Pi_1 \wedge \Pi_2$$

(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:

$$\begin{pmatrix} e_{14}, e_{15}, e_{16}, e_{1\alpha 2}, e_{24}, e_{25}, e_{26}, e_{2\alpha 2}, e_{34}, \\
    e_{35}, e_{36}, e_{3\alpha 2}, e_{\alpha 14}, e_{\alpha 15}, e_{\alpha 16}, e_{\alpha 1\alpha 2} \end{pmatrix}$$

(28)

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

$$\begin{pmatrix} e_{1245}, e_{1246}, e_{124\alpha 2}, e_{1256}, e_{125\alpha 2}, e_{126\alpha 2}, e_{1345}, e_{1346}, e_{134\alpha 2}, e_{1356}, e_{135\alpha 2}, \\
e_{136\alpha 2}, e_{1\alpha 145}, e_{1\alpha 146}, e_{1\alpha 14\alpha 2}, e_{1\alpha 156}, e_{1\alpha 15\alpha 2}, e_{1\alpha 16\alpha 2}, e_{2345}, e_{2346}, e_{234\alpha 2}, \\
e_{2356}, e_{235\alpha 2}, e_{236\alpha 2}, e_{245\alpha 2}, e_{246\alpha 2}, e_{256\alpha 2}, e_{26\alpha 2}, e_{2\alpha 145}, e_{2\alpha 146}, e_{2\alpha 14\alpha 2}, e_{2\alpha 156}, e_{2\alpha 15\alpha 2}, e_{2\alpha 16\alpha 2}, e_{33o145}, \\
e_{33o146}, e_{33o14\alpha 2}, e_{33o156}, e_{33o15\alpha 2}, e_{33o16\alpha 2} \end{pmatrix}$$

(29)

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

$$Q_{DCGA} \wedge L$$

(30)
As the number of components of $Q_{DCGA}$ is 25 and the number of components of $L$ 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

<table>
<thead>
<tr>
<th>Feature</th>
<th>DCGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>point is on a quadric</td>
<td>725</td>
</tr>
<tr>
<td>tangent plane</td>
<td>541</td>
</tr>
<tr>
<td>quadric-line intersection</td>
<td>900</td>
</tr>
</tbody>
</table>

**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: $\{w_0, w_1, w_2, w_3\}$ (corresponding to the projective Geometric Algebra vectors), and a copy of this basis $\{w_0^*, w_1^*, w_2^*, w_3^*\}$ such that $w_i w_i^* = 0.5 + w_i \wedge w_i^*$, $\forall i \in \{0,1,2,3\}$. To have more details, we show the inner products between any basis vectors in Table 3.

Table 3: Inner product between DPGA basis vectors.

<table>
<thead>
<tr>
<th></th>
<th>$w_0$</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_0^*$</th>
<th>$w_1^*$</th>
<th>$w_2^*$</th>
<th>$w_3^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>$w_0^*$</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_3^*$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**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:

\[ \mathbf{p} = xw_0 + yw_1 + zw_2 + w_3 \]
\[ \mathbf{p}' = xw_0' + yw_1' + zw_2' + w_3' \]  \hfill (31)

Note that the dual definition denotes the fact that:

\[ w_i \cdot w_j = \frac{1}{2} \delta_{ij} \quad \forall i, j = 0, \ldots, 3 \]  \hfill (32)

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

**Quadratics**

Again, for a general quadric defined as:

\[ ax^2 + by^2 + cz^2 + dxy + eyz + fzx + gx + hy + iz + j = 0 \]  \hfill (33)

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

\[
Q_{DPGA} = 4aw_0^* \wedge w_0 + 4bw_1^* \wedge w_1 + 4cw_2^* \wedge w_2 + 4dw_3^* \wedge w_3 \\
+ 2d(w_0^* \wedge w_1 + w_1^* \wedge w_0) + 2e(w_0^* \wedge w_2 + w_2^* \wedge w_0) \\
+ 2f(w_1^* \wedge w_2 + w_2^* \wedge w_1) + 2g(w_0^* \wedge w_3 + w_3^* \wedge w_0) \\
+ 2h(w_1^* \wedge w_3 + w_3^* \wedge w_1) + 2i(w_2^* \wedge w_3 + w_3^* \wedge w_2) \]  \hfill (34)

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

\[ \mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}' = 0 \]  \hfill (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 \( \mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}' \):

\[
f_{DPGA} = \mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}' \\
= \left( 2axw_0 + dxw_1 + exw_2 + gxw_3 + 2byw_0 + fyw_2 + hyw_3 + 2czw_2 + ezw_0 + fzw_1 + izw_3 \right) \cdot \mathbf{p}' \\
+ \left( (2ax + dy + ez + g)xw_0 + (2by + dx + fz + h)yw_1 \right) \cdot \mathbf{p}' \\
+ \left( (2cz + ex + fy + i)zw_2 + (iz + gx + hy + 2j)w_3 \right) \cdot \mathbf{p}' \] \hfill (36)

The last inner product results in:

\[
f_{DPGA} = \mathbf{p} \cdot Q_{DPGA} \cdot \mathbf{p}' \\
= ax^2 + 0.5dxy + 0.5exz + 0.5gx + by^2 + 0.5dxy + 0.5fyz + 0.5hy \\
+ cz^2 + 0.5exz + 0.5fyz + 0.5iz + 0.5iz + 0.5gx + 0.5hy + j \] \hfill (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 \( \Pi^* \) to a quadric \( Q_{DPGA} \) at a dual point \( p^* \) as follows:

\[
\Pi^* = Q_{DPGA} \cdot p^*
\]

**Quadric-line intersection**

DPGA also supports the quadric-line intersection. Given a line defined as the primal and dual \( L = x_1 \land x_2 \) and \( L^* = x_1^* \land x_2^* \) and the quadratic surface bivector \( Q_{DPGA} \). The intersection bivector called \( P_p \) is defined as follows:

\[
P_p = (L^* \land Q_{DPGA} \land L) \cdot I
\]

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

\[
I = w_0 \land w_1 \land w_2 \land w_3 \land w_4 \land w_5 \land w_6 \land w_7
\]

**Complexity of some major operations of DPGA**

\( Q_{DPGA} \) has a total of 16 basis bivector components. For each basis bivector, 2 inner products are performed. Thus, the first product \( P \cdot Q_{DCGA} \) 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 \( Q_{DPGA} \) and a line \( L \) and \( L^* \). The line \( L^* \) is obtained by the outer product of two points \( x_1 \) and \( x_2 \) whose number of components is 4. Thus a line \( L \) has 6 components along the bivector basis:

\[
(w_0 \land w_1, w_0 \land w_2, w_0 \land w_3, w_1 \land w_2, w_1 \land w_3, w_2 \land w_3)
\]

and a line \( L^* \) has the following bivector basis components:

\[
(w_0^* \land w_1^*, w_0^* \land w_2^*, w_0^* \land w_3^*, w_1^* \land w_2^*, w_1^* \land w_3^*, w_2^* \land w_3^*)
\]

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 \( L^* \land Q_{DPGA} \) is \( 6 \times 16 = 96 \) outer products. The result is a 4-vector and may have components along the quad-vector basis:

\[
(w_0^* \land w_1^* \land w_2^* \land w_3^*, w_0^* \land w_1^* \land w_2^* \land w_4^*, w_0^* \land w_1^* \land w_3^* \land w_5^*, w_0^* \land w_2^* \land w_3^* \land w_6^*, w_1^* \land w_2^* \land w_3^* \land w_7^*, w_1^* \land w_2^* \land w_4^* \land w_7^*, w_1^* \land w_3^* \land w_5^* \land w_7^*, w_2^* \land w_3^* \land w_5^* \land w_7^*, w_2^* \land w_4^* \land w_5^* \land w_7^*, w_3^* \land w_4^* \land w_5^* \land w_7^*)
\]
Thus, the resulting entity has 16 components. Furthermore, the line 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 $p$ is the cost of the following product:

$$\pi^* = Q_{DPGA} \cdot p^*$$  \hspace{1cm} (44)

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

**QCGA of $G_{9,6}$** Let us evaluate the computational cost of checking whether a point is on a quadratic surface. $Q^*$ 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 components is 12. Thus, the product $x \cdot Q^*$ 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:

$$n_e = \left((x \cdot e_1)e_{\infty 1} + (x \cdot e_2)e_{\infty 2} + (x \cdot e_3)e_{\infty 3} + e_1 \cdot q^*\right)e_1 +$$
$$\left((x \cdot e_2)e_{\infty 2} + (x \cdot e_1)e_{\infty 1} + (x \cdot e_3)e_{\infty 6} + e_2 \cdot q^*\right)e_2 +$$
$$\left((x \cdot e_3)e_{\infty 3} + (x \cdot e_1)e_{\infty 5} + (x \cdot e_2)e_{\infty 6} + e_3 \cdot q^*\right)e_3. \hspace{1cm} (45)$$

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:

$$\pi^* = n_e + \frac{1}{3}(e_{\infty 1} + e_{\infty 2} + e_{\infty 3}) \sqrt{-2(e_{\infty 1} + e_{\infty 2} + e_{\infty 3}) \cdot x}. \hspace{1cm} (46)$$

This computation requires the computation of an inner product of a vector with 3 components $(e_1, e_2, e_3)$ 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:

$$c^* = q^* \wedge l^* \hspace{1cm} (47)$$
The number of components of \( q^* \) is 12 as already seen. In QCGA, we defined a line with the 6 Plücker coefficients as:

\[
l^* = 3\mathbf{m} \mathbf{l}_c + (e_{\alpha 3} + e_{\alpha 2} + e_{\alpha 1}) \wedge \mathbf{n} \mathbf{l}_c.
\] (48)

The number of components of both \( m \) and \( n \) is 3. The outer product \((e_{\alpha 3} + e_{\alpha 2} + e_{\alpha 1}) \wedge \mathbf{n} \mathbf{l}_c\) yields a copy of the 3 components of \( n \) along \( e_{\alpha 1}, e_{\alpha 2}, e_{\alpha 3} \) basis vectors. Thus, the number of components of \( l^* \) is \( 3 \times 3 = 12 \). Finally, the cost of the outer product between \( q^* \) and \( l^* \) 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 tangent 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.

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>DPGA</th>
<th>DCGA</th>
<th>QCGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>point is on a quadric</td>
<td>144</td>
<td>750</td>
<td>144</td>
</tr>
<tr>
<td>tangent plane</td>
<td>64</td>
<td>541</td>
<td>180</td>
</tr>
<tr>
<td>quadric-line intersection</td>
<td>192</td>
<td>300</td>
<td>144</td>
</tr>
</tbody>
</table>

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:

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

can be extracted easily.
5 DCGA reciprocal operators

This means defining reciprocal operators for DCGA:

\[ T^x = e_1 \land e_4 \quad T^y = e_2 \land e_5 \]
\[ T^z = e_3 \land e_6 \quad T^1 = e_{o1} \land e_{o2} \]  

along with the 6 following:

\[ T^x = (e_1 \land e_{o2} + e_{o1} \land e_4) \]
\[ T^y = (e_2 \land e_{o2} + e_{o1} \land e_5) \]
\[ T^z = (e_3 \land e_{o2} + e_{o1} \land e_6) \]
\[ T^{xy} = (e_1 \land e_5 + e_2 \land e_4) \]
\[ T^{xz} = (e_1 \land e_6 + e_3 \land e_4) \]
\[ T^{yz} = (e_3 \land e_5 + e_2 \land e_6) \]

These reciprocal operators verify the following properties:

\[ T^x \cdot T^x = 1, \quad T^y \cdot T^y = 1, \quad T^z \cdot T^z = 1, \quad T^{xy} \cdot T^{xy} = 1, \quad T^{xz} \cdot T^{xz} = 1, \quad T^{yz} \cdot T^{yz} = 1 \]  
\[ T^x \cdot T^x = 1, \quad T^y \cdot T^y = 1, \quad T^z \cdot T^z = 1, \quad T^{1} \cdot T^{1} = 1 \]  

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

\[ T^x \cdot q_{DCGA} = a, \quad T^y \cdot q_{DCGA} = b, \quad T^z \cdot q_{DCGA} = c, \quad T^{xy} \cdot q_{DCGA} = d, \]
\[ T^{xz} \cdot q_{DCGA} = e, \quad T^{yz} \cdot q_{DCGA} = f, \quad T^1 \cdot q_{DCGA} = g, \quad T^y \cdot q_{DCGA} = h \]

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

\[ a = T^x \cdot q_{DCGA}, \quad b = T^y \cdot q_{DCGA}, \quad c = T^z \cdot q_{DCGA}, \quad d = T^{xy} \cdot q_{DCGA}, \quad e = T^{xz} \cdot q_{DCGA}, \quad f = T^{yz} \cdot q_{DCGA}, \quad g = T^1 \cdot q_{DCGA}, \quad h = T^y \cdot q_{DCGA} \]

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 \( \hat{x} \) of DCGA that we define as:

\[ \hat{x} = \frac{x}{x \cdot (e_{o1} \land e_{o2})} \]

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

\[ x = \hat{x} \cdot e_1, \quad y = \hat{x} \cdot e_2, \quad z = \hat{x} \cdot e_3 \]
6 DPGA reciprocal operators

In a similar way, let us note $W$ reciprocal operators for DPGA

\[
\begin{align*}
Wx^2 &= w_0^* \wedge w_0 \\
Wy^2 &= w_1^* \wedge w_1 \\
Wz^2 &= 2w_2^* \wedge w_0 \\
Wxy &= 2w_3^* \wedge w_0 \\
Wxz &= 2w_3^* \wedge w_0 \\
Wyz &= 2w_3^* \wedge w_1
\end{align*}
\]

(57)

Again, the following properties hold:

\[
\begin{align*}
Wx^2 \cdot Wx^2 &= 1, \\
Wy^2 \cdot Wy^2 &= 1, \\
Wz^2 \cdot Wz^2 &= 1, \\
Wxy \cdot Wxy &= 1, \\
Wxz \cdot Wxz &= 1, \\
Wyz \cdot Wyz &= 1, \\
Wx \cdot Wx &= 1, \\
Wy \cdot Wy &= 1, \\
Wz \cdot Wz &= 1, \\
W1 \cdot W1 &= 1
\end{align*}
\]

(58)

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

\[
\begin{align*}
Wx^2 \cdot q_{DPGA} &= a, \\
Wy^2 \cdot q_{DPGA} &= b, \\
Wz^2 \cdot q_{DPGA} &= c, \\
Wxy \cdot q_{DPGA} &= d, \\
Wxz \cdot q_{DPGA} &= e, \\
Wyz \cdot q_{DPGA} &= f, \\
Wx \cdot q_{DPGA} &= g, \\
Wy \cdot q_{DPGA} &= h, \\
Wz \cdot q_{DPGA} &= i, \\
W1 \cdot q_{DPGA} &= j
\end{align*}
\]

(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 $w_3$ components (or $w_3^*$ 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:

\[
\begin{align*}
Qx^2 &= \frac{1}{2}e_{o1} \\
Qy^2 &= \frac{1}{2}e_{o2} \\
Qz^2 &= \frac{1}{2}e_{o3} \\
Qxy &= e_{o4} \\
Qxz &= e_{o5} \\
Qyz &= e_{o6} \\
Q^x &= e_1 \\
Q^y &= e_2 \\
Q^z &= e_3 \\
Q^1 &= e_{o1} + e_{o2} + e_{o3}
\end{align*}
\]

(60)

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

\[
\begin{align*}
Qx^2 \cdot q^* &= a, \\
Qy^2 \cdot q^* &= b, \\
Qz^2 \cdot q^* &= c, \\
Qxy \cdot q^* &= d, \\
Qxz \cdot q^* &= e, \\
Qyz \cdot q^* &= f, \\
Q^x \cdot q^* &= g, \\
Q^y \cdot q^* &= h, \\
Q^z \cdot q^* &= i, \\
Q^1 \cdot q^* &= j
\end{align*}
\]

(61)
The construction of a QCGA point was already explained in the previous section. The reciprocal operation requires the computation of the normalization point $\hat{x}$ of QCGA.

**Proposition 71** For QCGA point $x$, the normalization is merely computed through an averaging of $e_1, e_2, e_3$ components thus of $e_0$ component, namely as:

$$\frac{-x}{x \cdot e_0}$$  \hspace{1cm} (62)

*Proof.* A scale $\alpha$ on $x$ acts the same way on all null basis vectors of $x$:

$$\alpha x = \alpha x \cdot e_0 + \frac{1}{2} \alpha (x^2 e_{01} + y^2 e_{02} + z^2 e_{03}) + xyae_{04} + xzae_{05} + yzae_{06} + \alpha e_{01} + \alpha e_{02} + \alpha e_{03}$$  \hspace{1cm} (63)

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

$$\alpha x \cdot e_{01} = -\alpha$$  \hspace{1cm} (64)

$$\alpha x \cdot e_{02} = -\alpha$$

$$\alpha x \cdot e_{03} = -\alpha$$

Thus, if $\alpha \neq 0$:

$$\frac{-3\alpha x}{\alpha x \cdot (e_{01} + e_{02} + e_{03})} \cdot e_{01} = -\frac{3\alpha x}{-3\alpha} \cdot e_{01} = x \cdot e_{01} = 1$$  \hspace{1cm} (65)

A similar result is obtained with $e_{02}$ and $e_{03}$:

$$\frac{-3\alpha x}{\alpha x \cdot (e_{01} + e_{02} + e_{03})} \cdot e_{02} = x \cdot e_{02} = 1$$  \hspace{1cm} (66)

$$\frac{-3\alpha x}{\alpha x \cdot (e_{01} + e_{02} + e_{03})} \cdot e_{03} = x \cdot e_{03} = -1$$  \hspace{1cm} (67)

Thus, we checked that for any scaled points $x_1, x_2$:

$$\frac{x_1}{x_1 \cdot e_0} \cdot \frac{x_2}{x_2 \cdot e_0} = -\frac{1}{2} \|x_{1e} - x_{2e}\|^2$$  \hspace{1cm} (68)

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

$$x = \hat{x} \cdot e_1, \quad y = \hat{x} \cdot e_2, \quad z = \hat{x} \cdot e_3$$  \hspace{1cm} (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.
In terms of Geometric Algebra computations, first we compute the quadratic:

\[ q^* = (x_1 \wedge x_2 \wedge \cdots \wedge x_9 \wedge I_9^+)^* \]  

(70)

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

\[
q_{DPGA} = 4(Q^{x^2} \cdot q^*)w_0^* \wedge w_0 + 4(Q^{y^2} \cdot q^*)w_1^* \wedge w_1 + 4(Q^{z^2} \cdot q^*)w_2^* \wedge w_2 \\
+ 4(Q^1 \cdot q^*)w_3^* \wedge w_3 \\
+ 2(Q^{xy} \cdot q^*)(w_0^* \wedge w_1 + w_1^* \wedge w_0) \\
+ 2(Q^{xz} \cdot q^*)(w_0^* \wedge w_2 + w_2^* \wedge w_0) \\
+ 2(Q^{yz} \cdot q^*)(w_0^* \wedge w_3 + w_3^* \wedge w_0) \\
+ 2(Q^y \cdot q^*)(w_1^* \wedge w_3 + w_3^* \wedge w_1) \\
+ 2(Q^z \cdot q^*)(w_2^* \wedge w_3 + w_3^* \wedge w_2) \\
+ 2(Q^1 \cdot q^*)e_{\infty 1} + 2(Q^y \cdot q^*)e_{\infty 2} + 2(Q^z \cdot q^*)e_{\infty 3} \\
+ \frac{(W \cdot q_{DPGA})}{3}(e_{\infty 1} + e_{\infty 2} + e_{\infty 3}).
\]  

(71)

The rotation is now performed as follows:

\[ q_{DPGA} = Rq_{DPGA}R^{-1} \]  

(72)

The rotor \( R \) is defined as:

\[ R = \exp(\frac{1}{2}\theta w_i w_i^*) \]  

(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:

\[
q^* = -(2(W^{x^2} \cdot q_{DPGA})e_{\infty 1} + 2(W^{y^2} \cdot q_{DPGA})e_{\infty 2} + 2(W^{z^2} \cdot q_{DPGA})e_{\infty 3} \\
+ (W^{xy} \cdot q_{DPGA})e_{\infty 4} + (W^{xz} \cdot q_{DPGA})e_{\infty 5} + (W^{yz} \cdot q_{DPGA})e_{\infty 6}) \\
+ ((W^x \cdot q_{DPGA})e_1 + (W^y \cdot q_{DPGA})e_2 + (W^z \cdot q_{DPGA})e_3) \\
- \frac{(W \cdot q_{DPGA})}{3}(e_{\infty 1} + e_{\infty 2} + e_{\infty 3}).
\]

(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