International Mathematical Forum, 2, 2007, no. 20, 981 - 1005

Design of Algorithms of Robot Vision Using Conformal Geometric Algebra

Luis Falcón-Morales

Mathematics Department
Tecnológico de Monterrey
Guadalajara, Jalisco 45201, México
luis.eduardo.falcon@itesm.mx

Eduardo Bayro-Corrochano

Electrical Engineering and Computer Science Department
Centro de Investigación y de Estudios Avanzados
Guadalajara, Jalisco 44550, Mexico
edb@gdl.cinvestav.mx

Abstract

In this paper the authors will apply a mathematical system, the Conformal Geometric Algebra (CGA), to propose applications in computer vision and robotics. The CGA keeps our intuition and insight of the problem's geometry at hand, besides helping us to reduce the computational problems' burden.

Matrix and vector algebras, complex numbers, rigid and conformal transformations, Euclidean and projective spaces, to name some, are different mathematical tools to model and solve almost any problem in robotic vision. Now, CGA is a mathematical system where all those systems are embedded. We use this compact system following the Occam's razor philosophy that mathematical 'entities should not be multiplied unnecessarily'.

In this work we handle simulated and real tasks for perception-action systems, treated in a single and efficient way. The authors show that this framework can be of great advantage for applications using stereo vision, range data, laser, omnidirectional and odometry based systems.

Mathematics Subject Classification: Applied Mathematics

Keywords: Computer vision; Clifford geometric algebra; incidence algebra; 3D rigid motion; visually guided robotics.

1 Introduction

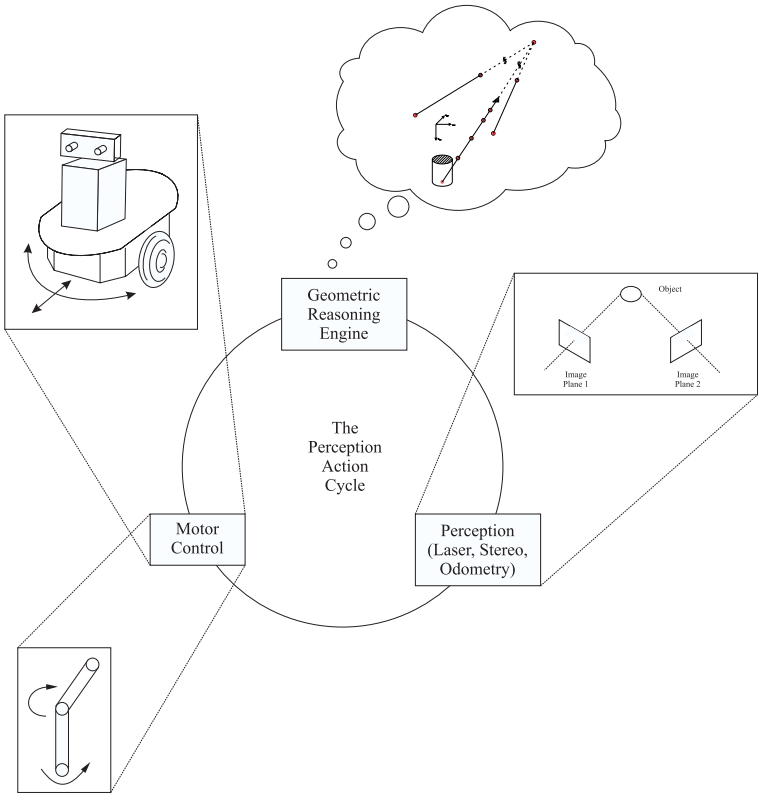

Figure 1: Abstraction of the perception action cycle.

In this paper we give an introduction of the proposal mathematical system CGA to the communities of applied mathematics, in particular computer vision and robotics, including not only the eulcidean and projective spaces but the operations of algebra of incidence, the directed distance and the rigid and conformal transformations as spinors.

Our mathematical approach appears promising for the development of perception action cycle systems, see Figure 1. This paper is an improvement to previous works [3, 4, 5, 6, 13], since using the CGA we are now including the group of transformations in our computations and expanding our study to more complex surfaces: ruled surfaces. Other authors have used Grassmann–Cayley algebra in computer vision [14] and robotics [19], but while they can express in this standard mathematical system the key ideas of projective geometry, such as the meet, join, duality and projective split, it lacks of an inner (contractive) product and of the group of transformations, which cannot be included in a very simple and natural way to the system.

In fact, in the 1960's David Hestenes take up again a proposal 'seeded' in

the XIX century about build a global mathematical framework, which would include the main mathematical systems of that era: matrices and determinants; vector calculus; complex numbers; conformal transformations; Euclidean and projective spaces; differential forms; differential geometry; ordinary and partial differential equations. Thus, CGA is the fusion of the Clifford Geometric Algebra (GA), and the non-Euclidean Hyperbolic Geometry. Historically, GA and CGA has not been taken into consideration seriously by the scientific community, but now and after the work of David Hestenes [10] and Pertti Lounesto [15] it has been taking a new scope of perspectives, not only theoretically, but for new and innovative applications to physics, computer vision, robotics and neural computing. One of the critics against CGA is the wrong idea that this system can manipulate only basic entities (points, lines, planes and spheres) and therefore it won't be useful to model general two and three dimensional objects, curves, surfaces or any other nonlinear entity required to solve a problem of a perception action system in robotics and computer vision.

In this paper, the authors put a lot of effort to explain clearly the CGA, illustrating the computations in great detail. Using the same ideas showed in this paper, another practical tasks of visual guided robotics could be implemented for 3D motion estimation, $body - eye$ calibration, 3D reconstruction, navigation, reaching and grasping 3D objects, etc. Thus, one of the main contributions of this paper is to introduce a suitable computational framework to the computer vision and robotic communities, which can be of great advantage for future applications in stereo vision, range data, laser, omnidirectional and odometry based systems.

Another contribution of this paper is the way that we show how to generate ruled surfaces or some other non trivial 3D curves using the CGA.

However, in this paper we present the CGA, with its *algebra of incidence* [12] and *rigid-motion* transformations, to obtain several practical techniques in the resolution of problems of perception action systems including ruled surfaces: 3D motion guidance of very non-linear curves; reaching and 3D object manipulation on very non-linear surfaces.

There are several interest points to study ruled surfaces: as robots and mechanisms are moving, any line attached to them will be tracing out a ruled surface or some other high nonlinear 3D-curve; the industry needs to guide the arm of robots with a laser welding to joint two ruled surfaces; reaching and manipulating 3D-objects is one of the main task in robotics, and it is usual that these objects have ruled surfaces or revolution surfaces; to guide a robot's arm over a critical 2D or 3D-curve or any other configuration constraint, and so forth.

The organization of the paper is as follows: section two presents a brief introduction to geometric algebra. Section three explains the conformal geometric algebra and directed distance and section four the rigid and conformal

transformations. In section five we present the way that several ruled surfaces or complex three dimensional curves can be generated in a very simple way using CGA. In section six we present simulated and real applications of the theory: to follow a 3D line, using the meet, that is the intersection of two planes or that is the intersection between ruled surfaces; grasping objects using directed distance. Finally, section seven presents the conclusions.

2 Geometric Algebra

The algebras of Clifford and Grassmann are well known to pure mathematicians, but since the beginning were abandoned by physicists in favor of the vector algebra of Gibbs, the commonly algebra used today in most areas of physics. The approach to Clifford algebra that we adopt here has been developed since the 1960's by David Hestenes [9], [10], [11], [12].

2.1 Basic definitions

Let V^n be a vector space of dimension n. We are going to define and generate an algebra G_n, called *geometric algebra*. Let $\{e_1, e_2, \ldots, e_n\}$ be a set of basis vectors of V^n. The *scalar multiplication* and *sum* in G_n are defined in the usual way of a vector space. The *product* or *geometric product* of elements of the basis of G_n will be simply denoted by juxtaposition. In this way, from any two basis vectors e_j and e_k, a new element of the algebra is obtained and denoted as $e_j e_k \equiv e_{jk}$. The product of basis vectors is anticommutative,

$$e_j e_k = -e_k e_j, \quad \forall j \neq k. \tag{1}$$

The basis vectors must *square* in $+1$, -1 or 0, this means that there are no-negative integers p, q and r such that $n = p + q + r$ and

$$e_i e_i = e_i^2 = \begin{cases} +1 & \text{for} \quad i = 1, \ldots, p \\ -1 & \text{for} \quad i = p+1, \ldots, p+q. \\ 0 & \text{for} \quad i = p+q+1, \ldots, n \end{cases} \tag{2}$$

This *product* will be called the *geometric product* of G_n. With these operations G_n is an associative linear algebra with identity and it is called the *geometric algebra* or *Clifford algebra* of dimension $n = p+q+r$, generated by the vector space V^n. It is usual to write $G_n \equiv G_{p,q,r}$.

The elements of this geometric algebra are called *multivectors*, because they are entities generated by the sum of elements of *mixed grade* of the basis set of G_n, such as

$$\mathbf{A} = \langle \mathbf{A} \rangle_0 + \langle \mathbf{A} \rangle_1 + \ldots + \langle \mathbf{A} \rangle_n, \tag{3}$$

where the multivector $\mathbf{A} \in G_n$ is expressed by the addition of its 0-*vector* part (scalar) $\langle \mathbf{A} \rangle_0$, its 1-*vector* part (vector) $\langle \mathbf{A} \rangle_1$, its 2-*vector* part (bivector) $\langle \mathbf{A} \rangle_2$, its 3-*vector* part (trivector) $\langle \mathbf{A} \rangle_3, \ldots$, and its n-*vector* part $\langle \mathbf{A} \rangle_n$. In particular, for $n = 3$, the standard basis of the geometric algebra G_3 is the following set, of $2^3 = 8$ elements,

$$\{1, e_1, e_2, e_3, e_{12}, e_{31}, e_{23}, e_{123}\}. \tag{4}$$

We call an r-*blade* or a *blade of grade* r to the geometric product of r linearly independent vectors.

It will be convenient to define other products between the elements of this algebra which will allow us to set up several geometric relations (unions, intersections, projections, etc.) between different geometric entities (points, lines, planes, spheres, etc.) in a very simple way.

Firstly, we define the *inner product*, $\mathbf{a} \cdot \mathbf{b}$, and the *exterior or wedge product*, $\mathbf{a} \wedge \mathbf{b}$, of any two 1-*vectors* $\mathbf{a}$, $\mathbf{b} \in G$, as the *symmetric* and *antisymmetric* parts of the geometric product $\mathbf{ab}$, respectively. That is, from the expression

$$\mathbf{ab} = \overbrace{\frac{1}{2}(\mathbf{ab} + \mathbf{ba})}^{symmetric\,part} + \overbrace{\frac{1}{2}(\mathbf{ab} - \mathbf{ba})}^{antisymmetric\,part} \tag{5}$$

we define the *inner product*

$$\mathbf{a} \cdot \mathbf{b} \equiv \frac{1}{2}(\mathbf{ab} + \mathbf{ba}) \tag{6}$$

and the *outer* or *wedge product*

$$\mathbf{a} \wedge \mathbf{b} \equiv \frac{1}{2}(\mathbf{ab} - \mathbf{ba}). \tag{7}$$

Thus, we can express now the geometric product of two vectors in terms of these two new operations as

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}. \tag{8}$$

From (6) and (7), $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$ and $\mathbf{a} \wedge \mathbf{b} = -\mathbf{b} \wedge \mathbf{a}$.
Now, we can define these two new operations for the more general elements of G_n, multivectors. For any two homogeneous multivectors $\mathbf{A}_r$ and $\mathbf{B}_s$ of grades r and s, respectively, we define the *inner product*

$$\mathbf{A}_r \cdot \mathbf{B}_s \equiv \begin{cases} \langle \mathbf{A}_r \mathbf{B}_s \rangle_{|r-s|}, & \text{if} \quad r > 0 \text{and } s > 0 \\ 0 & \text{if} \quad r = 0 \text{or } s = 0 \end{cases} \tag{9}$$

and the *outer* or *wedge product*

$$\mathbf{A}_r \wedge \mathbf{B}_s \equiv \langle \mathbf{A}_r \mathbf{B}_s \rangle_{r+s}. \tag{10}$$

We can see from equation (9), that the inner product of a scalar α with a homogeneous multivector $\mathbf{A}_r$ has a special treatment: $\mathbf{A} \cdot \alpha = \alpha \cdot \mathbf{A} = 0$. However, it is not the case for the outer product: $\mathbf{A} \wedge \alpha = \alpha \wedge \mathbf{A} = \langle \alpha \mathbf{A} \rangle_r \equiv \alpha \mathbf{A}$. We make the observation that software like GABLE or CLICAL has not this exceptional case for the inner product. Now, the inner and outer product of any two general multivectors will be obtained applying the left and right distributive laws over their homogeneous parts and then using equations (9) and (10). Note that the definition in (9) is not in contradiction with (6). Analogously, (10) and (7) are consistent.

From (9) it can be said that the inner product $\mathbf{A}_r \cdot \mathbf{B}_s$ *lowers the grade of* $\boldsymbol{A}_r$ *by* s *units* when $r \geq s > 0$, and from equation (10) that the outer product $\mathbf{A}_r \wedge \mathbf{B}_s$ *raises the grade of* $\boldsymbol{A}_r$ *by* s *units* for every r, $s \geq 0$.

Let us now extend the concept of *magnitude* to multivectors. For an homogeneous multivector $\mathbf{A}_r$ its *magnitude* is defined as $|\mathbf{A}_r| \equiv \sqrt{|\mathbf{A}_r \cdot \mathbf{A}_r|}$. Now, from (3) and $\mathbf{A}$ an arbitrary multivector, its *magnitude* is defined by the scalar

$$|\mathbf{A}| = \sqrt{\sum_{r=0}^{n} |\mathbf{A}_r|^2}. \tag{11}$$

Now, we can define the magnitude of a multivector $\mathbf{A}$ to be the positive scalar $|\mathbf{A}| = \left|\mathbf{A}^\dagger * \mathbf{A}\right|^{1/2}$, where the operation of *reversion* of a multivector $\mathbf{A} = \sum_{r=0}^{n} \mathbf{A}_r$, denoted $\mathbf{A}^\dagger$, is defined as

$$\mathbf{A}^\dagger = \sum_{r=0}^{n} (-1)^{\frac{r(r-1)}{2}} \mathbf{A}_r. \tag{12}$$

In particular, for an *r-vector* $\mathbf{A}_r$ of the form $\mathbf{A}_r = \mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_r$: $\mathbf{A}_r^\dagger = (\mathbf{a}_1 \cdots \mathbf{a}_{r-1}\mathbf{a}_r)^\dagger = \mathbf{a}_r \mathbf{a}_{r-1} \cdots \mathbf{a}_1$ and thus $\mathbf{A}_r^\dagger \mathbf{A}_r = \mathbf{a}_1^2\ \mathbf{a}_2^2 \cdots \mathbf{a}_r^2$, so, we will say that such a r-vector is null if and only if it has a null vector for a factor. If in such *factorization* of $\mathbf{A}_r$ p, q and s factors square in a positive number, negative and zero, respectively, we will say that $\mathbf{A}_r$ is a r-vector with signature (p, q, s). In particular, if $s = 0$ such a *non-singular* r-vector has a multiplicative inverse

$$\mathbf{A}^{-1} = (-1)^q \frac{\mathbf{A}^\dagger}{|\mathbf{A}|^2} = \frac{\mathbf{A}}{\mathbf{A}^2} \tag{13}$$

In general, the *inverse* $\mathbf{A}^{-1}$ of a multivector $\mathbf{A}$, if it exists, is defined by the equation $\mathbf{A}^{-1}\mathbf{A} = 1$.

Let G_n the geometric algebra spanned by the real vector space V^n. Because the addition of k-vectors (homogeneous vectors of grade k) is closed and the multiplication of a k-vector by a scalar is another k-vector, the set of all

k-vectors is a vector space, denoted $\bigwedge^k V^n$. Each of this spaces is spanned by $\begin{pmatrix} n \\ k \end{pmatrix}$ k-vectors, where $\begin{pmatrix} n \\ k \end{pmatrix} := \frac{n!}{(n-k)!k!}$. Thus, our geometric algebra G_n, which is spanned by $\sum_{k=0}^{n} \begin{pmatrix} n \\ k \end{pmatrix} = 2^n$ elements, is a direct sum of its homogeneous subspaces of grades 0, 1, 2, ..., n, that is,

$$G_n = \bigwedge^0 V^n \oplus \bigwedge^1 V^n \oplus \bigwedge^2 V^n \oplus \cdots \oplus \bigwedge^k V^n \oplus \cdots \oplus \bigwedge^n V^n \tag{14}$$

where $\bigwedge^0 V^n = R$ is the set of real numbers and $\bigwedge^1 V^n = V^n$.

Thus, any multivector of G_n can be expressed in terms of the basis of these subspaces. For instance, in G_3 a *typical* multivector u will be of the form

$$\mathbf{u} = \alpha_0 + \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3 + \alpha_4 e_{12} + \alpha_5 e_{31} + \alpha_6 e_{23} + \alpha_7 I_3 \tag{15}$$

where the α_i's are real numbers. When a subalgebra is generated only by multivectors of even grade, it is called the *even* subalgebra of G_n and will be denoted as G_n^+. Note however that the set of odd multivectors is not a subalgebra of G_n.

In $G_{p,\,q,\,r}$ we have the *pseudoscalar* $I_n \equiv e_{12\cdots n}$, but we can even define a *pseudoscalar* for each set of vectors that square in a positive number, negative number and zero, that is, $I_p \equiv e_{12\cdots p}$, $I_q \equiv e_{(p+1)(p+2)\cdots(p+q)}$ and $I_r \equiv e_{(p+q+1)(p+q+2)\cdots(p+q+r)}$ where $n = p + q + r$. We say that $G_{p,\,q,\,r}$ is degenerated if $r \neq 0$.

The manipulation of multivectors is easier with the use of the next equality for the inner product of two blades $\mathbf{A}_r = \mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_r$, and $\mathbf{B}_s = \mathbf{b}_1 \wedge \mathbf{b}_2 \wedge \cdots \wedge \mathbf{b}_s$

$$\mathbf{A}_r \cdot \mathbf{B}_s = \begin{cases} ((\mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_r) \cdot \mathbf{b}_1) \cdot (\mathbf{b}_2 \wedge \mathbf{b}_3 \wedge \cdots \wedge \mathbf{b}_s) & \text{if} \quad r \geq s \\ (\mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_{r-1}) \cdot (\mathbf{a}_r \cdot (\mathbf{b}_1 \wedge \mathbf{b}_2 \wedge \cdots \wedge \mathbf{b}_s)) & \text{if} \quad r < s \end{cases} \tag{16}$$

where

$$(\mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_r) \cdot \mathbf{b}_1 = \sum_{i=1}^{r} (-1)^{r-i} \, \mathbf{a}_1 \wedge \cdots \wedge \mathbf{a}_{i-1} \wedge (\mathbf{a}_i \cdot \mathbf{b}_1) \wedge \mathbf{a}_{i+1} \wedge \cdots \wedge \mathbf{a}_r \tag{17}$$

$$\mathbf{a}_r \cdot (\mathbf{b}_1 \wedge \mathbf{b}_2 \wedge \cdots \wedge \mathbf{b}_s) = \sum_{i=1}^{s} (-1)^{i-1} \, \mathbf{b}_1 \wedge \cdots \wedge \mathbf{b}_{i-1} \wedge (\mathbf{a}_r \cdot \mathbf{b}_i) \wedge \mathbf{b}_{i+1} \wedge \cdots \wedge \mathbf{b}_s. \tag{18}$$

From these equations we can see that the inner product is not commutative for general multivectors. Indeed, if in particular $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are 1-vectors, then from (18)

$$\mathbf{a} \cdot (\mathbf{b} \wedge \mathbf{c}) = (\mathbf{a} \cdot \mathbf{b})\mathbf{c} - (\mathbf{a} \cdot \mathbf{c})\mathbf{b}. \tag{19}$$

and from (17)

$$(\mathbf{b} \wedge \mathbf{c}) \cdot \mathbf{a} = \mathbf{b}(\mathbf{c} \cdot \mathbf{a}) - (\mathbf{b} \cdot \mathbf{a})\mathbf{c} = -(\mathbf{a} \cdot \mathbf{b})\mathbf{c} + (\mathbf{a} \cdot \mathbf{c})\mathbf{b}. \tag{20}$$

The *dual* of a multivector $\mathbf{A}$ in G_n is defined by

$$\mathbf{A}^* = \mathbf{A}I_n^{-1} \tag{21}$$

where I_n^{-1} differs from I_n by at most a sign.

2.2 The geometric algebra of 3-D space

The basis for the geometric algebra $\mathcal{G}_{3,0,0}$ of the 3-D space has $2^3 = 8$ elements and is given by:

$$\underbrace{1}_{scalar}, \underbrace{\{e_1, e_2, e_3\}}_{vectors}, \underbrace{\{e_{23}, e_{31}, e_{12}\}}_{bivectors}, \underbrace{\{e_{123}\} \equiv I}_{trivector}. \tag{22}$$

It can easily be verified that the trivector or pseudoscalar $\sigma_1\sigma_2\sigma_3$ squares to -1 and commutes with all multivectors in the 3-D space. We therefore give it the symbol I; noting that this is not the uninterpreted commutative scalar imaginary j used in quantum mechanics and engineering.

Multiplication of the three basis vectors e_1, e_2, and e_3 by I results in the three basis bivectors $e_{12} = Ie_3$, $e_{23} = Ie_1$ and $e_{31} = Ie_2$. These simple bivectors rotate vectors in their own plane by 90°, e.g. $(e_{12})e_2 = e_1$, $(e_{23})e_2 = -e_3$, etc. Identifying the $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ of the quaternion algebra with $I\sigma_1, -I\sigma_2, I\sigma_3$ the famous Hamilton relations $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ can be recovered. Since the $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are bivectors, it comes as no surprise that they represent 90° rotations in orthogonal directions and provide a well-suited system for the representation of general 3D rotations.

In geometric algebra a rotor (short name for rotator), $\mathbf{R}$, is an even-grade element of the algebra which satisfies $\mathbf{R}\widetilde{\mathbf{R}} = 1$, where $\widetilde{\mathbf{R}}$ stands for the *conjugate* of $\mathbf{R}$. That is, for a *r-vector* $\mathbf{A} = \sum_{i=0}^{r} \langle \mathbf{A} \rangle_i$ its *conjugate* is defined as

$$\widetilde{\mathbf{A}} = \sum_{i=0}^{r} (-1)^{\frac{i(i+1)}{2}} \langle \mathbf{A} \rangle_i. \tag{23}$$

If $\mathcal{A} = \{a_0, a_1, a_2, a_3\} \in \mathcal{G}_{3,0,0}$ represents a unit quaternion, then the rotor which performs the same rotation is simply given by

$$\begin{aligned} \mathbf{R} &= \underbrace{a_0}_{scalar} + \underbrace{a_1(Ie_1) - a_2(Ie_2) + a_3(Ie_3)}_{bivectors} \\ &= a_0 + a_1e_2e_3 - a_2e_3e_1 + a_3e_1e_2. \end{aligned} \tag{24}$$

The quaternion algebra is therefore seen to be a subset of the geometric algebra of 3-space. From (23) the conjugate of the rotor is $\widetilde{\mathbf{R}} = a_0 - a_1 e_{23} + a_2 e_{31} - a_3 e_{12}$.

The transformation in terms of a rotor $\mathbf{a} \mapsto \mathbf{R}\mathbf{a}\widetilde{\mathbf{R}} = \mathbf{b}$ is a very general way of handling rotations; it works for multivectors of any grade and in spaces of any dimension in contrast to quaternion calculus. Rotors combine in a straightforward manner, i.e. a rotor $\mathbf{R}_1$ followed by a rotor $\mathbf{R}_2$ is equivalent to a total rotor $\mathbf{R}$ where $\mathbf{R} = \mathbf{R}_2\mathbf{R}_1$.

A multivector $\mathbf{A} \in G_n$ is called *homogeneous* of grade r if $\mathbf{A} = \langle \mathbf{A} \rangle_r$.

The *dual* of a multivector $A \in G_n$ is defined by

$$\mathbf{A}^* = \mathbf{A}\ \mathbf{I}_n^{-1} \tag{25}$$

where $\mathbf{I}_n \equiv e_{12\cdots n}$ is the unit *pseudoscalar* of G_n and the *inverse* of a multivector $\mathbf{A}_n$, if it exists, is defined by the equation $\mathbf{A}^{-1}\mathbf{A} = 1$.

Duality give us the opportunity to define the *meet* $M \vee N$ between two multivectors M and N, using each of the following equivalent expressions

$$meet(M, N) \equiv M \vee N = M^* \cdot N = M^* \wedge N^*. \tag{26}$$

Geometrically, this operation will give us the intersection between geometric primitives through the intersection of their generated subspaces. See [12].

3 Conformal Geometric Algebra

The geometric algebra of a 3D Euclidean space $\mathcal{G}_{3,0,0}$ has a point basis and the motor algebra $\mathcal{G}_{3,0,1}$ a line basis. In the latter the lines expressed in terms of Plücker coordinates can be used to represent points and planes as well, [18], [7]. In CGA the unit element is the sphere, which will allow us to represent other entities. We begin this section giving an introduction to the conformal geometric algebra following the same formulation presented in [11] and [3], and showing how the Euclidean vector space $\mathbb{R}^n$ can be embedded in $\mathbb{R}^{n+1,1}$. For a gentle introduction to conformal geometric algebra the reader can resort to [8].

Let $\mathbb{R}^{n+1,1}$ be the vector space with an orthonormal vector basis given by $\{e_1, .., e_n, e_+, e_-\}$, with the property (2) expressed as:

$$e_i^2 = 1,\ \ e_+^2 = 1,\ \ e_-^2 = -1, \tag{27}$$

$$e_i \cdot e_+ = e_i \cdot e_- = e_+ \cdot e_- = 0 \tag{28}$$

for $i = 1, \ldots, n$. To simplify notation we are not going to write this basis in bold.

A *null basis* $\{e_0, e_\infty\}$ can be introduced by

$$e_0 = \frac{1}{2}(e_- - e_+), \tag{29}$$

$$e_\infty = e_- + e_+. \tag{30}$$

with the properties

$$e_0^2 = e_\infty^2 = 0,\ e_\infty \cdot e_0 = -1. \tag{31}$$

A unit pseudoscalar $\mathbf{E} \in \mathbb{R}^{1,1}$, representing the *Minkowski plane*, is defined by $\mathbf{E} \equiv e_\infty \wedge e_0 = e_+ \wedge e_- = e_+ e_-$. From here $\mathbf{E}^2 = 1$.

One of the results of the non-Euclidean geometry demonstrated by Nikolai Lobachevsky in the XIX century is that in spaces with hyperbolic structure we can find subsets which are isomorphic to a Euclidean space. In order to do this, Lobachevsky introduced two constraints, to the now called *conformal point* $\boldsymbol{x}_c \in \mathbb{R}^{n+1,1}$. See Figure 2. The first constraint is the *homogeneous* representation, *normalizing* the vector $\boldsymbol{x}_c$ such that

$$\boldsymbol{x}_c \cdot e_\infty = -1, \tag{32}$$

and the second constraint is such that the vector must be a *null vector*, that is,

$$\boldsymbol{x}_c^2 = 0. \tag{33}$$

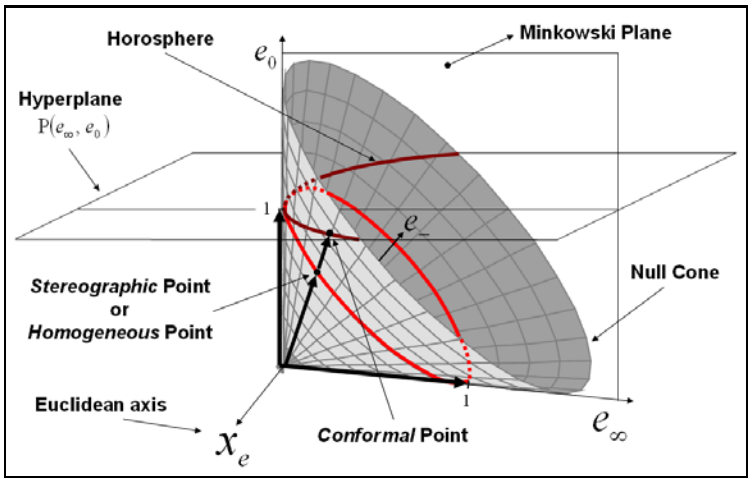

Figure 2: The Null Cone and the Horosphere for 1-D, and the conformal and stereographic representation of a 1-D vector.

Thus, conformal points are required to lie in the intersection surface, denoted $\mathbb{N}_e^n$, between the *null cone* $\mathbb{N}^{n+1}$ and the *hyperplane* $\mathbb{P}(e_\infty, e_0)$:

$$\begin{aligned} \mathbb{N}_e^n &= \mathbb{N}^{n+1} \cap \mathbb{P}(e_\infty, e_0) \\ &= \{\mathbf{x}_c \in \mathbb{R}^{n+1,1} | \mathbf{x}_c^2 = 0, \mathbf{x}_c \cdot e_\infty = -1\}. \end{aligned} \tag{34}$$

The constraint (34) define an isomorphic mapping between the Euclidean and the Conformal space. Thus, for each conformal point $\boldsymbol{x}_c \in \mathbb{R}^{n+1,1}$ there is a unique Euclidean point $\boldsymbol{x}_e \in \mathbb{R}^n$ and unique scalars α, β such that the mapping $\boldsymbol{x}_e \mapsto \boldsymbol{x}_c = \boldsymbol{x}_e + \alpha e_0 + \beta e_\infty$ is bijective. From (32) and (33) we can obtain $\alpha = 1$ and $\beta = \frac{1}{2}\boldsymbol{x}_e^2$. Then, the *standard form* of a conformal point $\boldsymbol{x}_c$ is

$$\boldsymbol{x}_c = \mathbf{x}_e + \frac{1}{2}\mathbf{x}_e^2 e_\infty + e_0. \tag{35}$$

3.1 Directed Distance

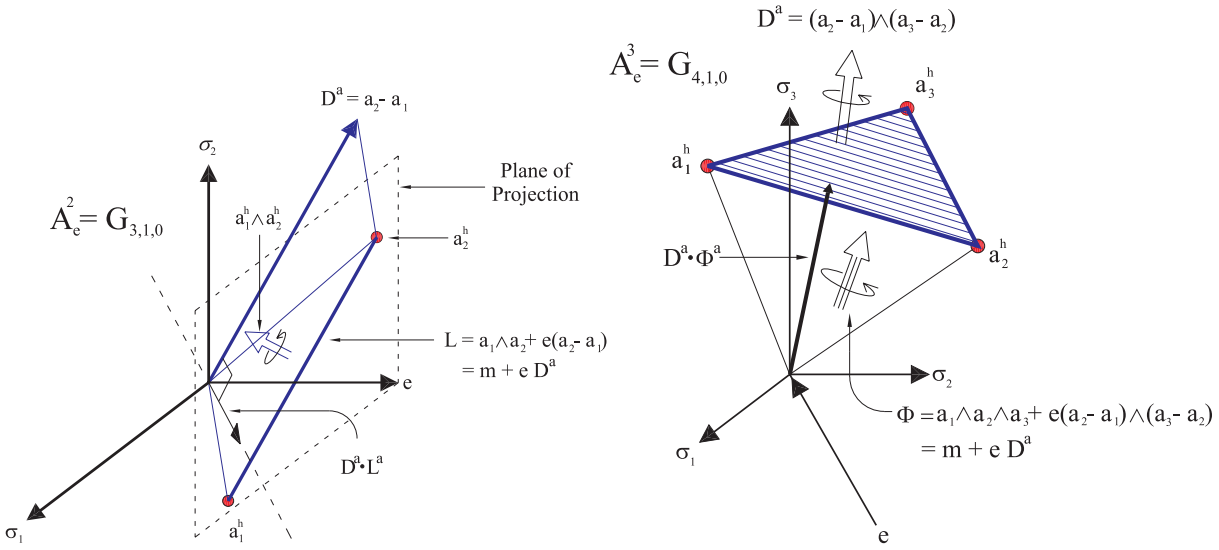

Figure 3: a) Line in 2D affine space. b) Plane in the 3D affine space (note that the 3D space is "lifted" by a null vector e .

It is well known from vector analysis the so-called *Hessian normal form*, a convenient representation to specify lines and planes using their distance from the origin (the Hesse distance or Directed distance). In this section we are going to show how CGA can help us to obtain the Hesse distance for more general simplexes and not only for lines and planes. Figure 3(a) and (b) depict a line and a plane, respectively, that will help us to develop our equations. Let $\mathbf{A}^k$ be a k-line (or plane), then it consist of a *momentum* M^k of degree k and of a *direction* D^{k-1} of degree $k-1$. For instance, given three Euclidean points a_1, a_2, a_3 their 2-simplex define a dual 3-plane in CGA that can be expressed as

$$\mathbf{A}^k \equiv \Phi = M^3 + D^2 e_0 = a_1 \wedge a_2 \wedge a_3 + (a_2 - a_1) \wedge (a_3 - a_1) e_0. \tag{36}$$

Then, the directed distance of this plane, denoted as $\mathbf{p}^k$, can be obtained taking the inner product between the unit direction D_u^{k-1} and the moment M^k. Indeed, from (36) and using expressions (27) to (31), we get the direction from $\Phi \cdot e_\infty = D^{k-1}$ and then its unitary expression D_u^{k-1} dividing D^{k-1} by its magnitude. Schematically,

$$\mathbf{A}^k \longrightarrow \mathbf{A}^k \cdot e_\infty = \mathbf{D}^{k-1} \longrightarrow \mathbf{D}_u^{k-1} = \frac{\mathbf{D}^{k-1}}{\left|\mathbf{D}^{k-1}\right|}. \tag{37}$$

Finally the directed distance $\mathbf{p}^k$ of $\mathbf{A}^k$ is

$$\mathbf{p}^k = \mathbf{D}_u^{k-1} \cdot \mathbf{A}^k, \tag{38}$$

where the dot operation basically takes place between the direction $\mathbf{D}_u^{k-1}$ and the momentum of $\mathbf{A}^k$. Obviously, the directed distance vector p^k touches orthogonally the k-plane $\mathbf{A}^k$, and as we mentioned at the beginning of this subsection, the magnitude $\left|\mathbf{p}^k\right|$ equals the Hesse distance. For sake of simplicity, in Figures (3.a) and (3.b) only $\mathbf{D}^{k-1} \cdot \mathbf{L}^k$ and $\mathbf{D}^{k-1} \cdot \Phi^k$ are respectively shown.

Now, having this point from the first object, we can use it to compute the directed distance from the k-plane $\mathbf{A}^k$ parallel to the object $\mathbf{B}^k$ as follows

$$d[\mathbf{A}^k, \mathbf{B}^k] = d[\mathbf{D}^{k-1} \cdot \mathbf{A}^k, \mathbf{B}^k] = d[(e_\infty \cdot \mathbf{A}^k) \cdot \mathbf{A}^k, \mathbf{B}^k]. \tag{39}$$

4 Direct and Opposite Motion Maps

In the middle of the XIX century J. Liouville proved, for the 3-dimensional case, that any conformal mapping on the whole of $\mathbb{R}^n$ can be expressed as a composition of *inversions in spheres* and *reflections in hyperplanes*, [16]. In particular, *rotation, translation, dilation* and *inversion* mappings will be obtained with these two mappings. CGA simplifies this concepts because the isomorphism between the conformal group on $\mathbb{R}^n$ and the Lorentz group on $\mathbb{R}^{n+1}$ help us to express, with a linear Lorentz transformation, a non-linear conformal transformation, and then to use *versor* representation to simplify *composition of transformations* with *multiplication of vectors*, [11]. Thus, using CGA is computationally more efficient and simpler to interpret the geometry of the conformal mappings, than with matrix algebra. A transformation of geometric figures is said to be *conformal* if it preserves the *shape* of the figures, that is, whether it preserves the angles and hence the shapes of straight lines and circles. In particular, rotation and translation mappings are conformal and are also called *direct-motion* transformations. Inversion and reflection mappings preserve the magnitude of the angle but reverse its direction and then they are also called *opposite-motion* transformations.

From [11] a conformal transformation in the geometric algebra framework uses a versor representation as

$$g(\mathbf{x}_c) = \mathbf{G}\mathbf{x}_c(\tilde{\mathbf{G}})^{-1} = \sigma\mathbf{x'}_c, \tag{40}$$

where $\mathbf{x}_c \in \mathbb{R}^{n+1,1}$, $\mathbf{G}$ is a *versor* (a multivector that can be expressed as the geometric product of r-invertible vectors), $\tilde{\mathbf{G}} = (-1)^r\mathbf{G}$ is the *main involution* of $\mathbf{G}$ and σ is a scalar. $\mathbf{G}$ can be expressed in geometric algebra as a composite of versors: *inversions* to spheres and *reflections* to hyperplanes.

4.1 Inversion with respect to a sphere

By the classical definition, an *inversion T_S with respect to a sphere S* (of radius ρ and centre $\mathbf{c}$) is such that for any point q at a distance d from $\mathbf{c}$, $T_S(q)$ will be in the same ray from $\mathbf{c}$ to q and at a distance ρ^2/d from $\mathbf{c}$. Figure 4(a). We comment some of the main properties of this transformation: (a) The inverse of a plane through the center of inversion is the plane itself. (b) The inverse of a plane not passing through the center of inversion is a sphere passing through the center of inversion. (c) The inverse of a sphere through the center of inversion is a plane not passing through the center of inversion. (d) The inverse of a sphere not passing through the center of inversion is a sphere not passing through the center of inversion. Figure 4(b). (e) Inversion in a sphere maps lines and circles to lines and circles.

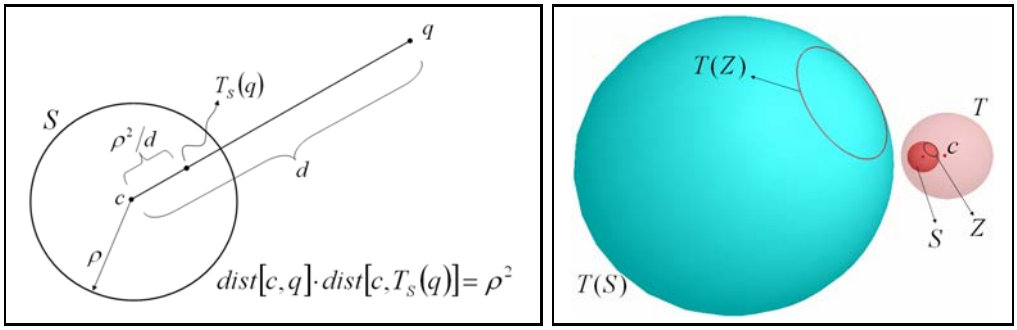

Figure 4: (a) The point $T_S(q)$ is the inverse point of q with respect to the circle S, and vice versa. (b) The inversion with respect to a sphere T mapping sphere to sphere and circle to circle, if they are not passing through the center of T.

In the context of the conformal geometry, the general form of a reflection about a vector s is, from 40,

$$g(\mathbf{x}_c) \equiv s(\mathbf{x}_c) = -\mathbf{s}\mathbf{x}_c\mathbf{s}^{-1} = \mathbf{x}_c - 2(\mathbf{s}\cdot\mathbf{x}_c)\mathbf{s}^{-1} = \sigma\mathbf{x'}_c, \tag{41}$$

where $\mathbf{s}\mathbf{x} + \mathbf{x}\mathbf{s} = 2(\mathbf{s}\cdot\mathbf{x})$, using the definition of the Clifford product between two vectors. The last equality of (41) is because $\mathbf{x}_c - 2(\mathbf{s}\cdot\mathbf{x}_c)\mathbf{s}^{-1}$ is a conformal

point that now is not normalized as in (32). Thus, σ is the scalar which get the normalization back again.

Now, in the conformal geometric framework the versor $\mathbf{s}$ in (40) can be *any* versor, in particular a sphere! Then using (40) we can obtain the inversion-reflexion of an entity with respect to a sphere and not only with respect to a single vector.

From [11] we know that in CGA the equation of a sphere of radius ρ centered at point $\mathbf{c}_c$ is the conformal vector

$$\mathbf{s} = \mathbf{c}_c - \frac{1}{2}\rho^2 e_\infty. \tag{42}$$

4.2 Reflections

A hyperplane with unit normal $\mathbf{n}$ and signed distance δ from the origin in $\mathbb{R}^n$ can be represented by the vector

$$\mathbf{s} = \mathbf{n} + \delta e_\infty. \tag{43}$$

Inserting $\mathbf{s} \cdot \mathbf{x}_c = \mathbf{n} \cdot \mathbf{x}_e$ into Eq. (41), we find that

$$g(\mathbf{x}_e) = \mathbf{n}\mathbf{x}_e\mathbf{n}^\dagger + 2\delta\mathbf{n} = \mathbf{n}(\mathbf{x}_e - \delta\mathbf{n})\mathbf{n}^\dagger + \delta\mathbf{n}, \tag{44}$$

where $\mathbf{n}^\dagger = -s^{-1}$. This expression is equivalent to a reflection $\mathbf{n}\mathbf{x}_e\mathbf{n}^\dagger$ at the origin, translated by δ along the direction of $\mathbf{n}$. A point $\mathbf{c}_e$ is on the hyperplane when $\delta = \mathbf{n} \cdot \mathbf{c}_e$, thus the equation (43) can be written as

$$\mathbf{s} = \mathbf{n} + e_\infty \mathbf{n} \cdot \mathbf{c}_e. \tag{45}$$

Via Eq. (44), this vector represents the reflection in a hyperplane through point $\mathbf{c}_e$.

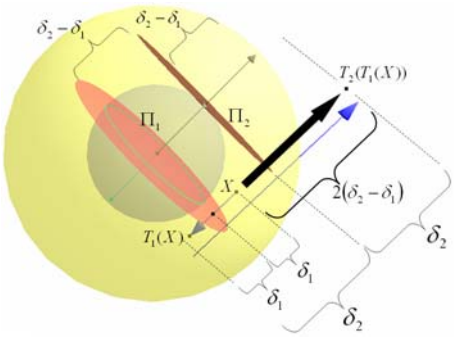

Figure 5: The translation from X to $T_2(T_1(X))$ is obtained as the inversion-reflection with two parallel planes. $\prod_1$ passes through the origin.

4.3 Translations

Translations can be modeled by two reflections, one reflection with respect to a plane passing through the origin and the other to a distance δ from the origin. Then, from Eq. (43), we can represent the operator for a translation (called a translator) as

$$\begin{aligned} \mathbf{T_a} = \pi_1\pi_2 &= (\mathbf{n} + \delta e_\infty)(\mathbf{n} + 0e_\infty), \\ &= 1 + \frac{1}{2}\mathbf{a}e_\infty, \end{aligned} \tag{46}$$

where $\mathbf{a} = 2\delta\mathbf{n}$ and $\mathbf{n}^2 = 1$. The translation distance is twice the separation between the hyperplanes. See Figure 5.

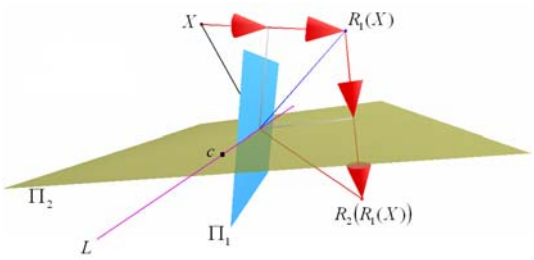

Figure 6: The rotation as the inversion-reflection with two non-parallel planes.

4.4 Rotations

Rotations can be modeled by the composition of two reflections about two hyperplanes intersecting in a common point $\mathbf{c}_e$, as in

$$\mathbf{R} = (\mathbf{a} + e_\infty \mathbf{a}\cdot\mathbf{c}_e)(\mathbf{b} + e_\infty \mathbf{b}\cdot\mathbf{c}_e) = \mathbf{ab} + e_\infty \mathbf{c}_e \cdot (\mathbf{a}\wedge\mathbf{b}), \tag{47}$$

where $\mathbf{a}$ and $\mathbf{b}$ are unit normal vectors. See Figure 6. Rotations about the origin can also be written in exponential form, as in

$$\mathbf{R} = e^{\frac{1}{2}\alpha\mathbf{B}}, \tag{48}$$

where $\mathbf{B}$ is a unit bivector, such that the dual $\mathbf{B}^*$ is the axis of rotation and α is the magnitude of the angle of rotation.

In general we can write down a canonical decomposition of a motor $\mathbf{M}$ for a rigid motion in terms of two versors, translation and rotation, as

$$\mathbf{M} = \mathbf{T_a}\mathbf{R}_\alpha. \tag{49}$$

5 Ruled Surfaces

Conics, ellipsoids, helicoids, hyperboloid of one sheet are entities which can not be directly described in CGA, however, can be modeled with its multivectors. In particular, a ruled surface is a surface generated by the displacement of a straight line (called generatrix) along a directing curve or curves (called directrices). The plane is the simplest ruled surface, but now we are interested in nonlinear surfaces generated as ruled surfaces. For example, a circular cone is a surface generated by a straight line through a fixed point and a point in a circle. It is well known that the intersection of a plane with the cone can generate the conics. See Figure 7. In [17] the cycloidal curves can be generated by two coupled twists. In this section we are going to see how these and other curves and surfaces can be obtained using only multivectors of CGA.

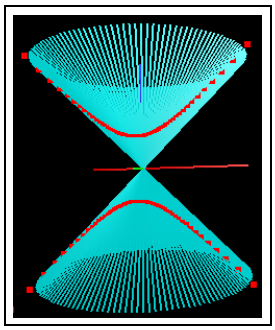

Figure 7: Hyperbola as the *meet* of a cone and a plane.

5.1 Cone and Conics

A circular cone is described by a fixed point v_0 (*vertex*), a dual circle $z_0 = a_0 \wedge a_1 \wedge a_2$ (*directrix*) and a rotor $\mathbf{R}(\theta, l)$, $\theta \in [0, 2\pi)$ rotating the straight line $L(v_0, a_0) = v_0 \wedge a_0 \wedge e_\infty$, (*generatrix*) along the axis of the cone $l_0 = z_0 \cdot e_\infty$. Then, the cone w is generated as

$$w = \mathbf{R}(\theta, l_0)\, L(v_0, a_0) \tilde{\mathbf{R}}(\theta, l_0)\,,\ \theta \in [0, 2\pi) \tag{50}$$

A conic curve can be obtained with the *meet* of a cone and a plane. See Figure 7.

5.2 Cycloidal Curves

The family of the cycloidal curves can be generated by the rotation and translation of one or two circles. For example, the cycloidal family of curves generated by two circles of radius r_0 and r_1 are expressed by, see Figure 8, the motor

$$\mathbf{M} = \mathbf{T}\mathbf{R}_1\mathbf{T}^*\mathbf{R}_2 \tag{51}$$

where

$$\mathbf{T} = \mathbf{T}((r_0 + r_1)(sin(\theta)e_1 + cos(\theta)e_2)) \tag{52}$$

$$\mathbf{R}_1 = \mathbf{R}_1(\frac{r_0}{r_1}\theta) \tag{53}$$

$$\mathbf{R}_2 = \mathbf{R}_2(\theta) \tag{54}$$

Then, each conformal point x is transformed as $\mathbf{M}\mathbf{x}\tilde{\mathbf{M}}$.

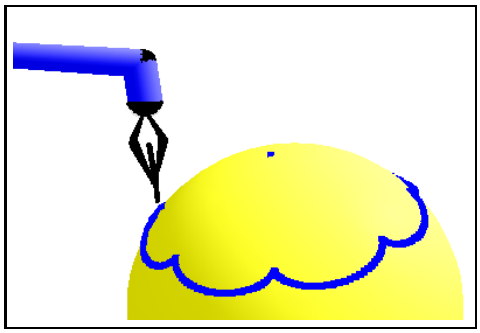

Figure 8: A laser welding following a 3D-curve: the projection of a cycloidal curve over a sphere.

5.3 Helicoid

We can obtain the ruled surface called helicoid rotating a ray segment in a similar way as the spiral of Archimedes. So, if the axis e_3 is the directrix of the rays and it is orthogonal to them, then the translator that we need to apply is a multiple of θ, the angle of rotation. See Figure 9.

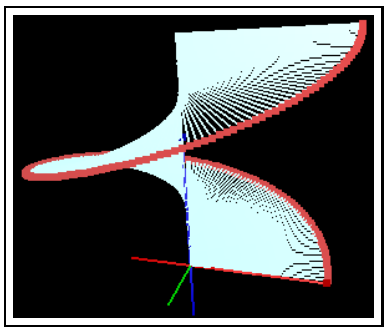

Figure 9: The helicoid is generated by the rotation and translation of a line segment. In CGA the *motor* is the desired multivector.

5.4 Sphere and Cone

Let us see an example of how the algebra of incidence using CGA simplify the algebra. The intersection of a cone and a sphere in general position, that is, the axis of the cone does not pass through the center of the sphere, is the three dimensional curve of all the Euclidean points (x, y, z) such that x and y satisfy the quartic equation

$$[x^2(1+\frac{1}{c^2}) - 2x_0x + y^2(1+\frac{1}{c^2}) - 2y_0y + x_0^2 + y_0^2 + z_0^2 - r^2]^2 = 4z_0^2(x^2+y^2)/c^2 \tag{55}$$

and x, y and z the quadratic equation

$$(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = r^2. \tag{56}$$

See Figure 10. In CGA the set of points q of the intersection can be expressed as the meet of the dual sphere s and the cone w, (50), defined in terms of its generatrix L, that is

$$q = (s^*) \cdot [\mathbf{R}\,(\theta, l_0)\, L(v_0, a_0)\tilde{\mathbf{R}}\,(\theta, l_0)],\ \theta \in [0, 2\pi)\,. \tag{57}$$

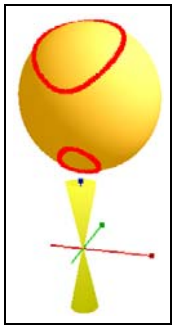

Figure 10: Intersection as the *meet* of a sphere and a cone.

Thus, in CGA we only need (57) to express the intersection of a sphere and a cone, meanwhile in Euclidean geometry it is necessary to use (55) and (56).

5.5 Hyperboloid of one sheet

The rotation of a line over a circle can generate a hyperboloid of one sheet. Figure 11.

5.6 Ellipse and Ellipsoid

The ellipse is a curve of the family of the cycloid and with a translator and a dilator we can obtain an ellipsoid.

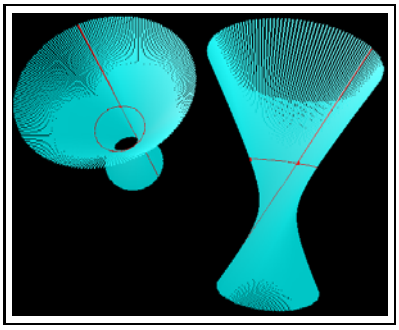

Figure 11: Hyperboloid as the rotor of a line.

5.7 Plücker Conoid

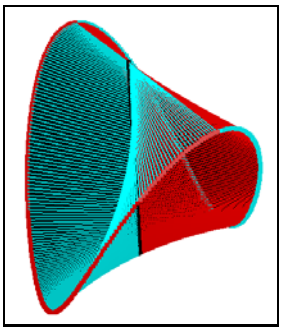

Figure 12: The Plücker conoid as a ruled surface.

The cylindroid or Plücker conoid is a ruled surface. See Figure 12. This ruled surface is like the helicoid where the translator parallel to the axis e_3 is of magnitude, a multiple of $cos(\theta)sin(\theta)$.

6 Applications

6.1 Line of intersection of two planes

In the industry, mainly in the sector dedicated to car assembly, it is often required to weld pieces. However, due to several factors, these pieces are not always in the same position, complicating this task and making this process almost impossible to automate. In many cases the requirement is to weld pieces of straight lines when no points on the line are available. This is the problem to solve in the following experiment.

Since we do not have the equation of the line or the points defining this line we are going to determine it via the intersection of two planes (the welding planes). In order to determine each plane, we need three points. The 3D

coordinates of the points are triangulated using the stereo vision system of the robot yielding a configuration like the one shown in Figure 13.

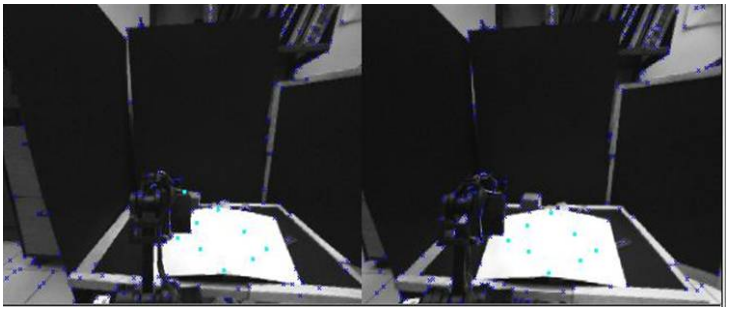

Figure 13: Images acquired by the binocular system of the robot "Geometer" showing the points on each plane.

Once the 3D coordinates of the points in space have been computed, we can find now each plane as $\pi^* = \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3 \wedge e_\infty$, and $\pi'^* = \mathbf{x}'_1 \wedge \mathbf{x}'_2 \wedge \mathbf{x}'_3 \wedge e'_\infty$. The line of intersection is computed via the *meet* operator $\mathbf{l} = \pi' \cap \pi$. In Figure 14.a we show a simulation of the arm following the line produced by the intersection of these two planes.

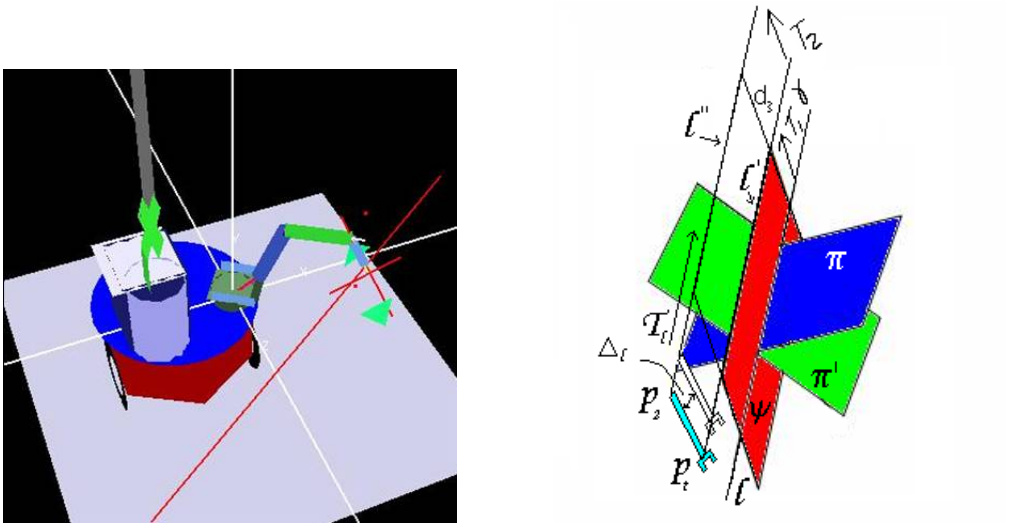

Figure 14: a) Simulation of the arm following the path of a line produced by the intersection of two planes. b) Guiding lines for the robotic arm produced by the intersection (meet) of planes and vertical translation.

Once the line of intersection $\mathbf{l}$ is computed, it suffices with translating it on the plane $\psi = \mathbf{l}^* \wedge e_2$ (see Figure 14.b) using the translator $\mathbf{T}_1 = 1 + \gamma e_2 e_\infty$, in the direction of e_2 (the y axis) a distance γ. Furthermore, we build the translator $\mathbf{T}_2 = 1 + d_3 e_2 e_\infty$ with the same direction (e_2), but with a separation

d_3 which corresponds to the size of the gripper. Once the translators have been computed, we find the lines $\mathbf{l}'$ and $\mathbf{l}''$ by translating the line $\mathbf{l}$ with $\mathbf{l}' = \mathbf{T}_1\mathbf{l}\mathbf{T}_1^{-1}$, and $\mathbf{l}'' = \mathbf{T}_2\mathbf{l}'\mathbf{T}_2^{-1}$.

The next step after computing the lines, is to find the points $\mathbf{p}_t$ and $\mathbf{p}_2$ which represent the places where the arm will start and finish its motion, respectively. These points were given manually, but they may be computed with the intersection of the lines $\mathbf{l}'$ and $\mathbf{l}''$ with a plane that defines the desired depth. In order to make the motion over the line, we build a translator $\mathbf{T}_L = 1 - \Delta_L \mathbf{l} e_\infty$ with the same direction as l as shown in Figure 14.b. Then, this translator is applied to the points $\mathbf{p}_2 = \mathbf{T}_L\mathbf{p}_2\ \mathbf{T}_L^{-1}$ and $\mathbf{p}_t = \mathbf{T}_L\mathbf{p}_t\mathbf{T}_L^{-1}$ in an iterative fashion to yield a displacement Δ_L on the robotic arm.

By placing the end point over the lines and p_2 over the translated line, and by following the path with a translator in the direction of l we get a motion over l as seen in the image sequence of Figure 15.

Figure 15: Image sequence of a linear-path motion.

6.2 Following a 3D-curve in ruled surfaces

As another simulated example using ruled surfaces consider a robot arm laser welder. See Figure 16. The welding distance has to be kept constant and the end-effector should follow a 3D-curve w on the ruled surface guided only by the directrices d_1, d_2 and a guide line L. From the generatrices we can always generate the nonlinear ruled surface, and then with the meet with another surface we can obtain the desired 3D-curve. We tested our simulations with several ruled surfaces, obtaining expressions of high nonlinear surfaces and 3D-curves, that with the standard vector and matrix analysis it would be very difficult to obtain them.

6.3 Grasping objects

Next, we will present an interesting task of grasping objects. First, we consider only approximately cubic objects (i.e., objects with nearly the same width, length, and height). We begin with four non-coplanar points belonging to the corners of the object and use them to build a sphere. With this sphere, we

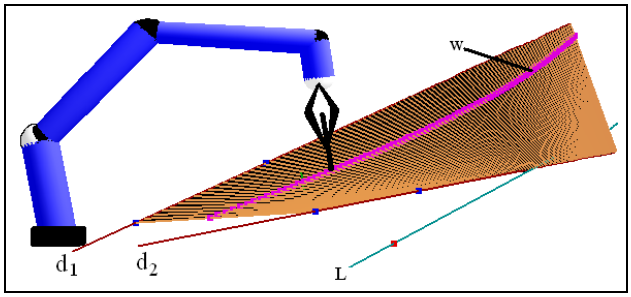

Figure 16: A laser welding following a 3D-curve w on a ruled surface defined by the directrices d_1 and d_2. The 3D-curve w is the *meet* between the ruled surface and a plane containing the line L.

can make either a horizontal or transversal section, so as to grasp the object from above, below, or in a horizontal fashion. Figure 17.a shows the sphere obtained using our simulator; the corners of the cube are shown in Figure 17.b; and Figure 17.c shows the robot arm moving its gripper toward the object after computing its inverse kinematics.

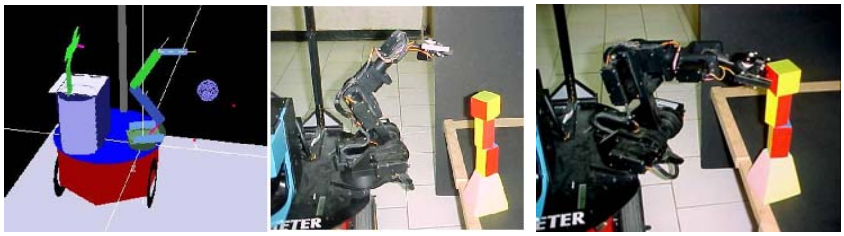

Figure 17: (a) Algorithm simulation showing the sphere containing the cube. (b) Image of the object we wish to grasp with the robotic arm. (c) The robot "Geometer" grasping a wooden cube.

The disadvantage of this algorithm is that not all the objects are cubes; therefore, we need to design a more general algorithm that allows the grasping of objects that are in the shape of regular prisms. That procedure is described next:

1. Take a calibrated stereo pair of images of the object.
2. Extract four non-coplanar points from these images (see, e.g., Figure 18.a).
3. Compute the corresponding 3-D points x_i, $i = 1, ..., 4$ using the stereo vision system and triangulation.
4. Compute the directed distances (see equation 38)

$$: d_1 \quad = \quad Dist(\mathbf{x}_1, \mathbf{x}_2 \wedge \mathbf{x}_3 \wedge \mathbf{x}_4 \wedge e_\infty),$$

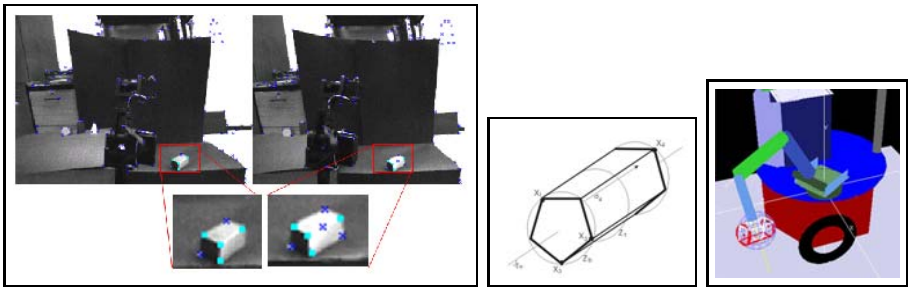

Figure 18: (a) Points on the object as seen by the robot. (b) Regular prism with height d_4 and main axis j_{z_b}. (c) Simulation of the algorithm showing the robotic arm grasping the object.

$$\begin{aligned} d_2 &= Dist(\mathbf{x}_2, \mathbf{x}_1 \wedge \mathbf{x}_3 \wedge \mathbf{x}_4 \wedge e_\infty), \\ d_3 &= Dist(\mathbf{x}_3, \mathbf{x}_2 \wedge \mathbf{x}_1 \wedge \mathbf{x}_4 \wedge e_\infty), \\ d_4 &= Dist(\mathbf{x}_4, \mathbf{x}_2 \wedge \mathbf{x}_3 \wedge \mathbf{x}_1 \wedge e_\infty). \end{aligned}$$

5. Select the point with the greatest distance as the apex x_a and label the rest $\mathbf{x}_{b_1}, \mathbf{x}_{b_2}, \mathbf{x}_{b_3}$ as belonging to the base of the object.

6. Compute the circle $\mathbf{z}_b = \mathbf{x}_{b_1} \wedge \mathbf{x}_{b_2} \wedge \mathbf{x}_{b_3}$.

7. Compute the directed distance $\mathbf{d}_a$ between $\mathbf{z}_b$ and $\mathbf{x}_a$.

8. Translate the circle $\mathbf{z}$ in the direction and magnitude of $\mathbf{d}_a$ to produce the grasping plane.

Some points of the previous algorithm can be explained in more detail. For example, for the object in Figure 18.b, the base circle is $\mathbf{z}_b^* = \mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \mathbf{x}_3$, whereas the main axis of the object is computed by $j_{z_b} = \mathbf{z}_b \wedge e_\infty$. The translator that moves $\mathbf{z}_b$ is produced as $\mathbf{T} = 1 + \frac{1}{4} d_4 e_\infty$. The grasping circle can be computed with $\mathbf{z}_t^* = \mathbf{T}\mathbf{z}_b^*\mathbf{T}^{-1}$, the point of contact being the closest point from the circle to the y-axis. Finally, the grasping plane is denoted by $\pi^* = \mathbf{z}_t^* \wedge e_\infty$. Note that this last algorithm may grasp the object regardless of whether it is in a horizontal or vertical position. We illustrate this algorithm with the simulation shown in Figure 18.c.

7 Conclusion

In this paper we have seen a single non–standard mathematical framework, the conformal geometric algebra, in order to simplify the set of data structures that we usually use with the traditional methods. The key idea is to define and use a set of products in CGA that will be enough to generate conformal

transformations, manifolds as ruled surfaces and develop incidence algebra operations, as well as solve equations and obtain directed distances between different kinds of geometric primitives. Thus, within this approach, all those different mathematical entities and tasks can be done now simultaneously, without the necessity of abandoning the novel system.

Real and simulated applications are proposed in order to show to the robotic and computer vision communities the useful insights and advantages of the CGA, and we invite them to adopt, explore and implement new tasks with this novel framework, expanding its horizon to new possibilities for robots equipped with stereo systems, range data, laser, omnidirectional and odometry.

Finally, remembering the *Occam's razor* principle, we prefer to use one single mathematical system instead of several of them to design, model and solve problems of robotic vision: *one should not increase, beyond what is necessary, the number of entities required to explain anything.*

References

[1] D.H. Ackeley, G.E. Hilton and T.J. Sejnovski, A learning algorithm for Bolzmann machine, *Cognitive Science,* **62** (1985), 147 - 169.

[2] D.O. Hebb, *Organization of Behaviour,* Wiley, New York, 1949.

[3] Bayro-Corrochano E. 2001. *Geometric Computing for Perception Action Systems*, Springer Verlag, Boston.

[4] Bayro-Corrochano, E. and Lasenby, J. 1995. Object modeling and motion analysis using Clifford algebra. In Proceedings of Europe-China Workshop on *Geometric Modeling and Invariants for Computer Vision*, Ed. Roger Mohr and Wu Chengke, Xi'an, China, April 27-29, pp. 143-149.

[5] Bayro-Corrochano, E., Lasenby, J. and Sommer, G. 1996. Geometric Algebra: a framework for computing point and line correspondences and projective structure using n–uncalibrated cameras. *Proceedings of the International Conference on Pattern Recognition (ICPR'96), Vienna, August 1996.*, Vol.I, pp. 393-397.

[6] Laseby J. and Bayro–Corrochano E. 1999. Analysis and Computation of Projective Invariants from Multiple Views in the Geometric Algebra Framework. In Special Issue on Invariants for Pattern Recognition and Classification, ed. M.A. Rodrigues. Int. Journal of Pattern Recognition and Artificial Intelligence, Vol 13, No 8, December 1999, pp. 1105–1121.

[7] Bayro–Corrochano E., Daniilidis K. and Sommer G. Motor algebra for 3D kinematics. The case of the hand–eye calibration. *Journal of Mathematical Imaging and Vision*, vol. 13, pag. 79-99, 2000.

[8] Bayro–Corrochano E. Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neurocomputing and Robotics, Springer-Verlag, Heidelberg, June 2005.

[9] Hestenes, D. 1966. Space–Time Algebra. *Gordon and Breach*.

[10] Hestenes, D. and Sobczyk, G. 1984. Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics. *D. Reidel*, Dordrecht.

[11] Hestenes D., Li H. and Rockwood A. 2001. New algebraic tools for classical geometry. *In Geometric Computing with Clifford Algebra*, G. Sommer (Ed.). Springer-Verlag, Berlin Heidelberg, Chap.I, pp. 3-23.

[12] Hestenes, D. and Ziegler, R. 1991. Projective Geometry with Clifford Algebra. *Acta Applicandae Mathematicae*, 23, pp. 25–63.

[13] Lasenby, J., Lasenby, A.N., Lasenby, Doran, C.J.L and Fitzgerald, W.J. 'New geometric methods for computer vision – an application to structure and motion estimation'. *International Journal of Computer Vision*, 26(3), 191-213. 1998.

[14] Csurka G. and Faugeras O. Computing three dimensional project invariants from a pair of images using the Grassmann–Cayley algebra *Journal of Image and Vision Computing*,

[15] Lounesto P. *Clifford Algebra and Spinors*, 2nd ed. Cambridge University Press, Cambridge, UK, 2001.

[16] Needham T. *Visual Complex Analysis*. Oxford University Press. Reprinted 2003.

[17] Rosenhahn B., Perwass C., Sommer G. *Pose estimation of 3D free-form contours*. Technical Report 0207, University Kiel, 2002.

[18] J.M. Selig. *Clifford algebra of points, lines and planes*. South Bank University, School of Computing, Information Technology and Maths, Technical Report SBU–CISM–99–06, 1999.

[19] White N. Geometric applications of the Grassmann–Cayley algebra. In J.E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*, CRC Press, Floridam 1997.

Received: September 6, 2006