Clifford Geometric Algebra: A Promising Framework for Computer Vision, Robotics and Learning

Eduardo Bayro-Corrochano

Computer Science Department, GEOVIS Laboratory
Centro de Investigación y de Estudios Avanzados
CINVESTAV, Guadalajara, Jalisco 44550, Mexico
edb@gdl.cinvestav.mx
http://www.gdl.cinvestav.mx/~edb

Abstract. In this paper the authors use the framework of geometric algebra for applications in computer vision, robotics and learning . This mathematical system keeps our intuitions and insight of the geometry of the problem at hand and it helps us to reduce considerably the computational burden of the problems. The authors show that framework of geometric algebra can be in general of great advantage for applications using stereo vision, range data, laser, omnidirectional and odometry based systems. For learning the paper presents the Clifford Support Vector Machines as a generalization of the real- and complex-valued Support Vector Machines.

1 What Is Clifford Geometric Algebra?

Let $\mathcal{G}_n$ denote the geometric algebra of n-dimensions – this is a graded linear space. As well as vector addition and scalar multiplication we have a non-commutative product which is associative and distributive over addition – this is the *geometric* or *Clifford product*. A further distinguishing feature of the algebra is that any vector squares to give a scalar. The geometric product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is written $\boldsymbol{ab}$ and can be expressed as a sum of its symmetric and anti-symmetric parts $\boldsymbol{ab} = \boldsymbol{a}\cdot\boldsymbol{b} + \boldsymbol{a}\wedge\boldsymbol{b}$. The outer or wedge product of two vectors is a new quantity which we call a *bivector*. We think of a bivector as a oriented area in the plane containing $\boldsymbol{a}$ and $\boldsymbol{b}$, formed by sweeping $\boldsymbol{a}$ along $\boldsymbol{b}$. The outer product is immediately generalizable to higher dimensions – for example, $(\boldsymbol{a}\wedge\boldsymbol{b})\wedge\boldsymbol{c}$, a *trivector*, is interpreted as the oriented volume formed by sweeping the area $\boldsymbol{a}\wedge\boldsymbol{b}$ along vector $\boldsymbol{c}$. The outer product of k vectors is a k-vector or k-blade, and such a quantity is said to have *grade* k. A *multivector* (linear combination of objects of different type) is *homogeneous* if it contains terms of only a single grade.

In an n-dimensional space we can introduce an orthonormal basis of vectors $\{\sigma_i\}$, $i = 1, ..., n$, such that $\sigma_i \cdot \sigma_j = \delta_{ij}$. This leads to a basis for the entire algebra:

$$1, \quad \{\sigma_i\}, \quad \{\sigma_i \wedge \sigma_j\}, \quad \{\sigma_i \wedge \sigma_j \wedge \sigma_k\}, \quad \ldots, \quad \sigma_1 \wedge \sigma_2 \wedge \ldots \wedge \sigma_n. \tag{1}$$

Note that the basis vectors are not represented by bold symbols. Any multivector can be expressed in terms of this basis. In this paper we will specify a geometric algebra $\mathcal{G}_n$ of the n dimensional space by $\mathcal{G}_{p,q,r}$, where p, q and r stand for the number of basis vector which squares to 1, -1 and 0 respectively and fulfill n=p+q+r. Its even subalgebra will be denoted by $\mathcal{G}^+_{p,q,r}$. For example $\mathcal{G}^+_{0,2,0}$ has the basis

$$\{1, \quad \sigma_1, \quad \sigma_2, \quad \sigma_1 \wedge \sigma_2\}, \tag{2}$$

where σ_1^2=-1,σ_2^2=-1. This means p=0, q=2 and r=0. Thus the dimension of this geometric algebra is n=$p+q+r$=2.

In the n-D space there are multivectors of grade 0 (scalars), grade 1 (vectors), grade 2 (bivectors), grade 3 (trivectors), etc... up to grade n. Any two such multivectors can be multiplied using the geometric product. Consider two multivectors $\boldsymbol{A}_r$ and $\boldsymbol{B}_s$ of grades r and s respectively. The geometric product of $\boldsymbol{A}_r$ and $\boldsymbol{B}_s$ can be written as

$$\boldsymbol{A}_r\boldsymbol{B}_s = \langle \mathbf{AB} \rangle_{r+s} + \langle \mathbf{AB} \rangle_{r+s-2} + \ldots + \langle \mathbf{AB} \rangle_{|r-s|} \tag{3}$$

where $\langle \boldsymbol{M} \rangle_t$ is used to denote the t-grade part of multivector $\boldsymbol{M}$, e.g. consider the geometric product of two vectors $\boldsymbol{ab} = \langle \boldsymbol{ab} \rangle_0 + \langle \boldsymbol{ab} \rangle_2 = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b}$.

For an detailed introduction to geometric algebra the reader should resort to [1–3].

2 Body-Eye Calibration

The so-called hand-eye calibration problem involves the computation of the transformation between a coordinate system attached to a robotic hand and the camera on top of it. Since we want calibrate a binocular head with a mobile robot, from know on we will call this task as the body-eye calibration problem.

The robot-to-sensor relation can be seen as a series of joints $J_1, J_2, ..., J_n$ (where a rotation about joint J_i affects all joints $J_{i+1}, ..., J_n$) and a measurement system U which is rigidly attached to the last joint J_n. The problem can be stated as the computation of the transformations $\boldsymbol{M}_1, \boldsymbol{M}_2, ..., \boldsymbol{M}_{n-1}$ between the robot frame and the last joint and the transformation $\boldsymbol{M}_n$ between the last joint and the measurement device U, using only data gathered with U. The procedure consists of two stages. The first stage computes the *screw axes* of the joints, and the second stage uses these axes to compute the final transformation between the coordinate systems.

2.1 Screw Axes Computation

To compute the axes of rotation, we use a motion estimator, for details see [2]. Each joint J_i is moved in turn while leaving the rest at their home position (see Figure 1.a) . From the resulting motor $\boldsymbol{M}_i$, the axis of rotation $\boldsymbol{S}_i$ can be extracted, , for details see [2]. For our particular robot, the sequence of motions is presented in Figure 1.b.

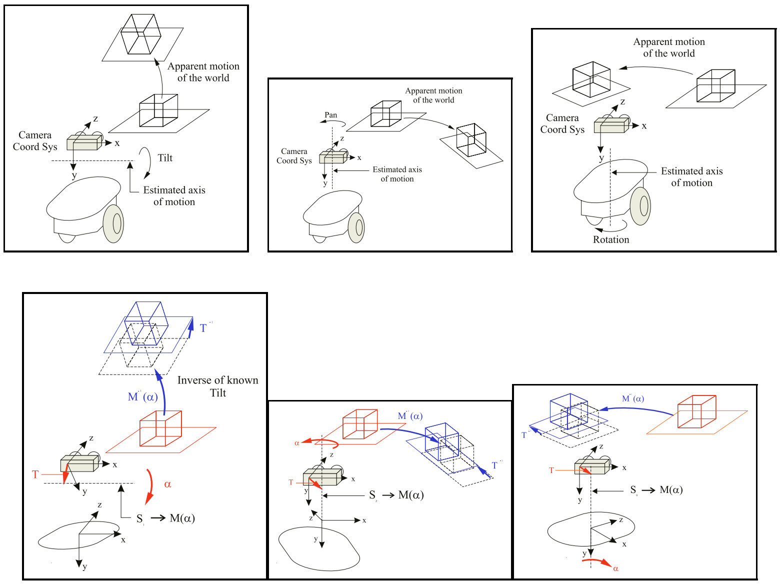

Fig. 1. (a) (*upper row*) Estimation of the screw axes. (b) (*lower row*) Correction of the rotation and relocation of the screw axes.

2.2 Calibration

Our algorithm will produce a set of lines $\boldsymbol{S}_i$ in the camera's coordinate system. Once these axes are known, the transformation taking one point $\boldsymbol{x}_k$ measured in the camera's framework to the robot's coordinate system is easy to derive, provided that we know the angles α_i applied to each joint J_i. Basically, the algorithm undoes the implicit transformations applied on the camera's framework by first rotating about joint J_k and then translating the joint (and the framework, along with the rest of the joints) to the origin (see Figure 1.b).

The functions used in our algorithm are defined as follows:

$$\texttt{nearest}(\boldsymbol{x}) = \frac{(\bar{e} \cdot \boldsymbol{x}) \cdot \boldsymbol{x}}{\bar{e} \cdot [(\bar{e} \cdot \boldsymbol{x}) \cdot \boldsymbol{x}]}, \tag{4}$$

$$\texttt{makeTranslator}(\mathbf{t}) = 1 + \frac{\boldsymbol{t}}{2}\bar{e}, \tag{5}$$

$$\texttt{lineToMotor}(\boldsymbol{L}^a, \alpha) = \cos(\frac{\alpha}{2}) + \sin(\frac{\alpha}{2})[\bar{e} \wedge (e_{123}\boldsymbol{m}) - (e_{123}\boldsymbol{n})], \tag{6}$$

where $\boldsymbol{L}^a = \boldsymbol{m} + e \wedge \boldsymbol{n}$, and $\boldsymbol{n} = 1$. The function $\texttt{nearest}(\boldsymbol{x})$ returns the point on $\boldsymbol{x}$ which is nearest to the origin, $\texttt{makeTranslator}(\mathbf{t})$ returns a translator displacing by an amount $\boldsymbol{t}$, and $\texttt{lineToMotor}(\boldsymbol{L}^a, \alpha)$, simply returns a motor that rotates α radians about the axis $\boldsymbol{L}^a$.

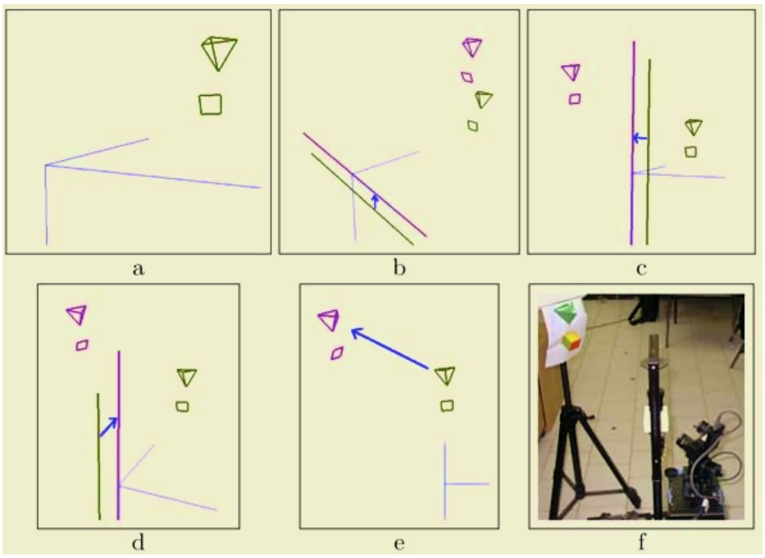

Fig. 2. (a) Reconstruction without calibration. (b–d) Relocation of the screws. (e) Comparison of the final reconstruction with the real view.

3 Inverse Kinematics and Object Manipulation

In this section we show how to perform certain object manipulation tasks in the context of conformal geometric algebra. First, we solve the inverse kinematics for a Pan-Tilt unit so that the binocular head will be able to follow the end-effector and to position the gripper of the arm in a certain position in space. Then, we we show how to grasp an object in space.

3.1 Inverse Kinematics for a Pan-Tilt Unit

In this task we apply a language of spheres for solving the inverse kinematics, this can be seen as an extension of an early approach [1], when a language of points, lines and planes were used instead.

In the inverse kinematics for a pan-tilt unit problem we aim to determine the angles θ_{tilt} and θ_{pan} of stereo-head, so that the cameras fix at the point p_t. We will now show how we find the values of θ_{pan} and θ_{tilt} using the conformal approach. The problem will be divided in three steps to be solved.

Step 1: Determine the point p_2.
When the θ_{tilt} rotes and the bases rotate (θ_{pan}) around the l_y (see Fig.3.a), the point p_2 describes a sphere s_1. This sphere has center at the point p_1 and radius d_2.

$$S_1 = p_1 - \frac{d_2^2}{2} e_\infty \tag{7}$$

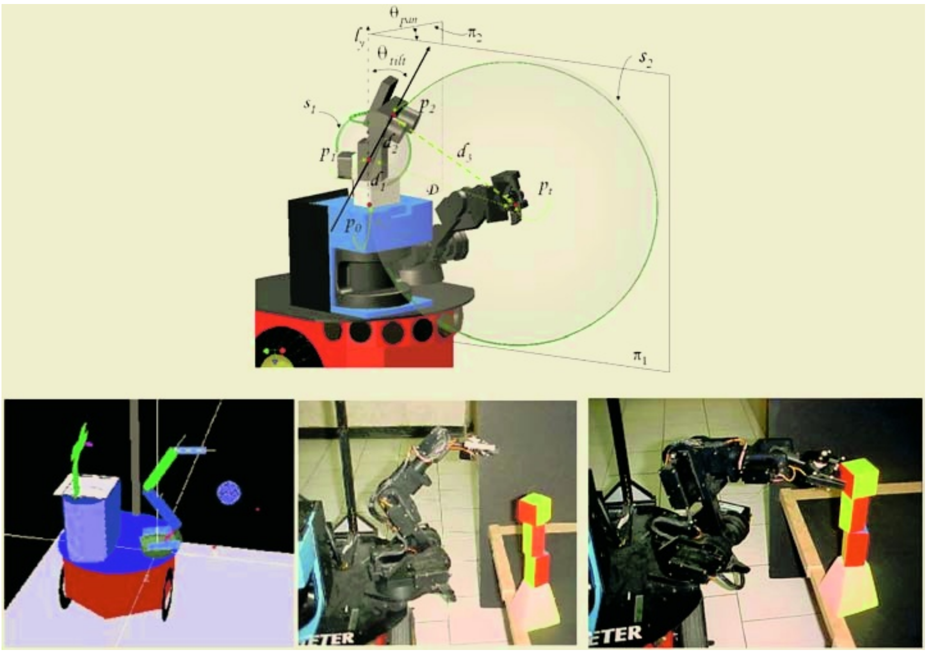

Fig. 3. a) Point p_2 given by intersection of the plane π_1 and the spheres s_1 and s_2. (b) Algorithm simulation showing the sphere containing the cube. (c) Image of the object we wish to grasp with the robotic arm. (d) The robot "Geometer" grasping a wooden cube.

Also the point p_t can be locked from every point around it. that is the point p_2 is in the sphere:

$$S_2 = p_t - \frac{d_3^2}{2} e_\infty \tag{8}$$

Where d_3 is the distance between point p_t and the cameras, and we can calculate d_3 using a Pythagorean theorem $d_3^2 = D^2 - d_2^2$, where D is the direct distance between p_t and p_1. We have restricted the position of the point p_2, but there is another restriction: the vector going from the p_2 to the point p_t must be live at the plane π_1 generated by the l_y axis ($l_y^* = p_0 \wedge p_1 \wedge e_\infty$) and the point p_t, as we can see in Fig. 3.a So that p_2 can be determined by intersecting the plane π_1 with the spheres s_1 and s_2 as follows

$$\pi_1^* = l_y^* \wedge p_t, \qquad P_{p2} = s_1 \wedge \pi_1 \wedge s_2. \tag{9}$$

Step 2: Determine the lines and planes.
Once p_2 have been determined, the line l_2 and the plane π_2 can be defined. This line and plane will be useful to calculate the angles θ_{tilt} and θ_{pan}.

$$l_2^* = p_1 \wedge p_2 \wedge e_\infty, \qquad \pi_2^* = l_y^* \wedge e_3. \tag{10}$$

Step 3: Find the angles θ_{tilt} and θ_{pan}.
Once we have all the geometric entities, the computation of the angles is a trivial step.

$$\cos(\theta_{pan}) = \frac{\pi_1^* \cdot \pi_2^*}{|\pi_1^*|\,|\pi_2^*|}, \qquad \cos(\theta_{tilt}) = \frac{l_1^* \cdot l_y^*}{|l_1^*|\,|l_y^*|}. \tag{11}$$

3.2 Grasping an Object

Other interesting experiments involve tasks 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 can make either a horizontal or transversal section, so as to grasp the object from above, below, or in a horizontal fashion. Figure 3.b shows the sphere obtained using our simulator; the corners of the cube are shown in Figure 3.c; and Figure 3.d shows the robot arm moving its gripper toward the object after computing its inverse kinematics.

4 Learning: Clifford Valued Support Vector Machines

In this section we will present the Clifford valued Support Vector Machine (CSVM). A CSVM will be a multivector generalization of the real and complex valued Support Vector Machines [4].

4.1 Linear Clifford Support Vector Machines for Classification

For the case of the Clifford SVM for classification we represent the data set in a certain Clifford Algebra G_n where $n = p + q + r$, where any multivector base squares to 0, 1 or -1 depending if they belong to p, r, or s multivector bases respectively. Each data *ith*-vector has multivector entries $\boldsymbol{x}_i = [\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{iD}]^T$, where $\boldsymbol{x}_{ij} \in \mathcal{G}_n$ and D is its dimension. Thus the *ith*-vector dimension is D$\times 2^n$. Each data *ith*-vector $\boldsymbol{x}_i \in \mathcal{G}_n^D$ of the N data vectors will be associated with their labels as follows: $(\boldsymbol{x}_{i1},\ \boldsymbol{y}_{i1})$,$(\boldsymbol{x}_{i2},\ \boldsymbol{y}_{i2})$,...,$(\boldsymbol{x}_{ij},\ \boldsymbol{y}_{ij})$,...,$(\boldsymbol{x}_{iD},\ \boldsymbol{y}_{iD})$, where each $\boldsymbol{y}_{ij} = y_{ij_s} + y_{ij_{\sigma_1}} + y_{ij_{\sigma_2}} + ... + y_{ij_I} \in \{\pm 1 \pm \sigma_1 \pm \sigma_2 ... \pm I\}$, where the first subindex s stands for scalar part. The 2^n classification problem is to separate these multivector-valued samples into 2^n groups by selecting a right function from the set of functions $\{f(\boldsymbol{x}) = \boldsymbol{w}^{*T}\boldsymbol{x} + \boldsymbol{b},\ \boldsymbol{x}, \boldsymbol{w} \in \mathcal{G}_n^D,\ \boldsymbol{b} \in \mathcal{G}_n^D$. The optimal weight vector will be

$$\boldsymbol{w} = [\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_D]^T \in \mathcal{G}_n^D. \tag{12}$$

Let us see in detail the last equation

$$\begin{aligned} f(\boldsymbol{x}) = \boldsymbol{w}^{*T}\boldsymbol{x} + \boldsymbol{b} &= [\boldsymbol{w}_1^*, \boldsymbol{w}_2^*, ..., \boldsymbol{w}_D^*][(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_D]^T + \boldsymbol{b} \\ &= \sum_{i=1}^{D} \boldsymbol{w}_i^* \boldsymbol{x}_i + \boldsymbol{b}, \end{aligned} \tag{13}$$

where $\boldsymbol{w}_i^*\boldsymbol{x}_i$ corresponds to the Clifford product of two multivectors and $\boldsymbol{w}_i^*$ is the conjugated of the multivector $\boldsymbol{w}$.

We introduce now a structural risk functional similar to the real valued one of the SVM and use loss function similar to the *function insensitive* ξ of Vapnik.

$$\begin{aligned}
&min \quad \frac{1}{2}\boldsymbol{w}^{*T}\boldsymbol{w} + C \cdot \sum_{j=1}^{l} \xi_i^2 && (14)\\
&subject\,to \quad Coef_s(y_{ij})Coef_s(f(\boldsymbol{x}_{ij})) \geq 1 - Coef_s(\xi_{ij})\\
&\qquad Coef_{\sigma_1}(y_{ij})Coef_{\sigma_1}(f(\boldsymbol{x}_{ij})) \geq 1 - Coef_{\sigma_1}(\xi_{ij})\\
&\qquad ...\\
&\qquad Coef_I(y_{ij})Coef_I(f(\boldsymbol{x}_{ij})) \geq 1 - Coef_I(\xi_{ij})\\
&Coef_s(\xi_{ij}) \geq 0,\ Coef_{\sigma_1}(\xi_{ij}) \geq 0,\ ...,\ Coef_I(\xi_{ij}) \geq 0,\ j = 1,\ ...,l,
\end{aligned}$$

where the subindex $i = 1, ..., D$.

The dual expression of this problem can be derived straightforwardly. Firstly let us consider the expression of the orientation of optimal hyperplane.

Since the $\boldsymbol{w}_i = [\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2}, ..., \boldsymbol{w}_{iD}]^T$, each of the $\boldsymbol{w}_{ij}$ is given by the multivector

$$\boldsymbol{w}_{ij} = w_{is} + w_{i\sigma_1}\sigma_1 + ... + w_{i\sigma_n}\sigma_n + w_{i\sigma_1\sigma_2}\sigma_1\sigma_2 + ... + w_{iI}I. \qquad (15)$$

Each component of these weights are computed as follows:

$$\begin{aligned}
w_{is} &= \sum_{j=1}^{l}\Big((\alpha_{is})_j(y_{is})_j\Big)(x_{is})_j,\ \ w_{i\sigma_1} = \sum_{j=1}^{l}\Big((\alpha_{i\sigma_1})_j(y_{i\sigma_1})_j\Big)(\boldsymbol{x}_{i\sigma_1})_j\ ...,\\
w_{iI} &= \sum_{j=1}^{l}\Big((\alpha_{iI})_j(y_{iI})_j\Big)(x_{iI})_j. \qquad (16)
\end{aligned}$$

According the Wolfe dual programing [4] the dual form reads

$$\min \quad \frac{1}{2}(\boldsymbol{w}^{*T}\boldsymbol{w}) - \sum_{j=1}^{l}\Big(\sum_{i=1}^{D}\Big((\alpha_{is})_j + ... + (\alpha_{i\sigma_1\sigma_2})_j + ... + (\alpha_{iI})_j\Big)\Big) \qquad (17)$$

subject to $\boldsymbol{a}^T \cdot \mathbf{1} = 0$, where the entries of the vector

$$\boldsymbol{a} = [\boldsymbol{a}_s, \boldsymbol{a}_{\sigma_1}, \boldsymbol{a}_{\sigma_2},\ ...,\ \boldsymbol{a}_{\sigma_1\sigma_2},\ \boldsymbol{a}_I] \qquad (18)$$

are given by

$$\begin{aligned}
\boldsymbol{a}_s^T = \Big[&[(\alpha_{1s1})(y_{1s_1}),\ (\alpha_{2s1})(y_{2s_1}),\ ...,\ (\alpha_{Ds1})(y_{Ds_1})],\ ...,\\
&[(\alpha_{1sl})(y_{1s_l}),\ (\alpha_{2sl})(y_{2s_l}),\ ...,\ (\alpha_{Dsl})(y_{Ds_l})]\Big],\\
\boldsymbol{a}_{\sigma_1}^T = \Big[&[(\alpha_{1\sigma_11})(y_{1\sigma_11}),\ (\alpha_{2\sigma_11})(y_{2\sigma_11}),\ ...,\ (\alpha_{D\sigma_11})(y_{D\sigma_11})],\ ...,
\end{aligned}$$

$$[(\alpha_{1\sigma_1 l})(y_{1\sigma_1 l}),\ (\alpha_{2\sigma_1 l})(y_{2\sigma_1 l}),\ ...,\ (\alpha_{D\sigma_1 l})(y_{D\sigma_1 l})]\Big]$$

$$\boldsymbol{a}_I^T = \Big[[(\alpha_{1I1})(y_{1I_1}),\ (\alpha_{2I1})(y_{2I_1}),\ ...,\ (\alpha_{DI1})(y_{DI_1})],\ ...,$$
$$[(\alpha_{1Il})(y_{1I_l}),\ (\alpha_{2Il})(y_{2I_l}),\ ...,\ (\alpha_{DIl})(y_{DI_l})]\Big], \tag{19}$$

note that each data *ith*-vector, $i = 1, ... N$, has D multivector entries and after the training we take into account not N but l *ith*-vectors which is the number of the found support vectors each one belonging to $\mathcal{G}_n^D$. Thus $\boldsymbol{a}^T$ has the dimension: $(D \times l) \times 2^n$, the latter multiplicand corresponds to the length of a multivector of $\mathcal{G}_n$.

In $\boldsymbol{a}^T \cdot \mathbf{1} = 0$, $\mathbf{1}$ denotes a vector of all ones, and all the Lagrange multipliers should fulfill $0 \leq (\alpha_{is})_j \leq C$, $0 \leq (\alpha_{i\sigma_1})_j \leq C$, ..., $0 \leq (\alpha_{i\sigma_1\sigma_2})_j \leq C$, ..., $0 \leq (i\alpha_I)_j \leq C$ for $i = 1, ..., D$ and $j = 1, ..., l$.

We require a compact an easy representation of the resultant *GRAM matrix* of the multi-components, this will help for the programing of the algorithm. For that let us first consider the Clifford product of $(\boldsymbol{w}^{*T}\boldsymbol{w})$, this can be expressed as follows

$$\boldsymbol{w}^*\boldsymbol{w} = \langle \boldsymbol{w}^*\boldsymbol{w}\rangle_s + \langle \boldsymbol{w}^*\boldsymbol{w}\rangle_{\sigma_1} + \langle \boldsymbol{w}^*\boldsymbol{w}\rangle_{\sigma_2} + \ldots + \langle \boldsymbol{w}^*\boldsymbol{w}\rangle_I. \tag{20}$$

Since $\boldsymbol{w}$ has the components presented in equation (16), the equation (20) can be rewritten as follows

$$\begin{aligned}\boldsymbol{w}^*\boldsymbol{w} &= \boldsymbol{a}^{*T}_s\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_s\boldsymbol{a}_s + ... + \boldsymbol{a}^{*T}_s\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_{\sigma_1\sigma_2}\boldsymbol{a}_{\sigma_1\sigma_2} + ... + \boldsymbol{a}^{*T}_s\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_I\boldsymbol{a}_I + \\ &\boldsymbol{a}^{*T}_{\sigma_1}\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_s\boldsymbol{a}_s + ... + \boldsymbol{a}^{*T}_{\sigma_1}\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_{\sigma_1\sigma_2}\boldsymbol{a}_{\sigma_1\sigma_2} + ... + \boldsymbol{a}^{*T}_{\sigma_1}\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_I\boldsymbol{a}_I + \\ &\boldsymbol{a}^{*T}_I\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_s\boldsymbol{a}_s + \boldsymbol{a}^T_I\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_{\sigma_1}\boldsymbol{a}_{\sigma_1} + ... + \boldsymbol{a}^{*T}_I\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_{\sigma_1\sigma_2}\boldsymbol{a}_{\sigma_1\sigma_2} + ... + \boldsymbol{a}^{*T}_I\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_I\boldsymbol{a}_I.\end{aligned} \tag{21}$$

Renaming the matrices of the t-grade parts of $\langle \boldsymbol{x}^*\boldsymbol{x}\rangle_t$, we rewrite previous equation as:

$$\begin{aligned}\boldsymbol{w}^*\boldsymbol{w} &= \boldsymbol{a}^{*T}_s\boldsymbol{H}_s\boldsymbol{a}_s + \boldsymbol{a}^{*T}_s\boldsymbol{H}_{\sigma_1}\boldsymbol{a}_{\sigma_1} + ... + \boldsymbol{a}^{*T}_s\boldsymbol{H}_{\sigma_1\sigma_2}\boldsymbol{a}_{\sigma_1\sigma_2} + ... + \boldsymbol{a}^{*T}_s\boldsymbol{H}_I\boldsymbol{a}_I + \\ &\boldsymbol{a}^{*T}_{\sigma_1}\boldsymbol{H}_s\boldsymbol{a}_s + \boldsymbol{a}^{*T}_{\sigma_1}\boldsymbol{H}_{\sigma_1}\boldsymbol{a}_{\sigma_1} + ... + \boldsymbol{a}^{*T}_{\sigma_1}\boldsymbol{H}_{\sigma_1\sigma_2}\boldsymbol{a}_{\sigma_1\sigma_2} + ... + \boldsymbol{a}^{*T}_{\sigma_1}\boldsymbol{H}_I\boldsymbol{a}_I + \\ &\boldsymbol{a}^{*T}_I\boldsymbol{H}_s\boldsymbol{a}_s + \boldsymbol{a}^{*T}_I\boldsymbol{H}_{\sigma_1}\boldsymbol{a}_{\sigma_1} + ... + \boldsymbol{a}^{*T}_I\boldsymbol{H}_{\sigma_1\sigma_2}\boldsymbol{a}_{\sigma_1\sigma_2} + ... + \boldsymbol{a}^{*T}_I\boldsymbol{H}_I\boldsymbol{a}_I.\end{aligned} \tag{22}$$

These results help us finally to rewrite equation (17) as a compact equation as follows

$$min \quad \frac{1}{2}\boldsymbol{w}^{*T}\boldsymbol{w} + C \cdot \sum_{j=1}^{l} \xi_i^2 = \frac{1}{2}\boldsymbol{a}^{*T}\boldsymbol{H}\boldsymbol{a} + C \cdot \sum_{j=1}^{l} \xi_i^2 \tag{23}$$

$$subject\,to \quad Coef_s(y_{ij})Coef_s(f(\boldsymbol{x}_{ij})) \geq 1 - Coef_s(\xi_{ij}), \tag{24}$$

where $\boldsymbol{a}$ is given by equation (18).

$\boldsymbol{H}$ is a positive semidefinite matrix which is the expected *Gramm* matrix. This matrix in terms of the matrices of the t-grade parts of $\langle \boldsymbol{x}^* \boldsymbol{x} \rangle_t$ is written as follows:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_s \boldsymbol{H}_{\sigma_1} \boldsymbol{H}_{\sigma_2} \ldots\ldots\ldots\ldots \boldsymbol{H}_{\sigma_1\sigma_2} \ldots \boldsymbol{H}_I \\ \boldsymbol{H}_{\sigma_1}^T \boldsymbol{H}_s \ldots\ \boldsymbol{H}_{\sigma_4} \ldots\ldots \boldsymbol{H}_{\sigma_1\sigma_2} \ldots \boldsymbol{H}_I \boldsymbol{H}_s \\ \boldsymbol{H}_{\sigma_2}^T \boldsymbol{H}_{\sigma_1}^T \boldsymbol{H}_s \ldots \boldsymbol{H}_{\sigma_1\sigma_2} \ldots \boldsymbol{H}_I \boldsymbol{H}_s \boldsymbol{H}_{\sigma_1} \\ . \\ . \\ . \\ \boldsymbol{H}_I^T \ldots \boldsymbol{H}_{\sigma_1\sigma_2}^T \ldots\ldots\ldots\ldots \boldsymbol{H}_{\sigma_2}^T \boldsymbol{H}_{\sigma_1}^T \boldsymbol{H}_s \end{bmatrix}, \tag{25}$$

note that the diagonal entries equal to $\boldsymbol{H}_s$ and since $\boldsymbol{H}$ is a symmetric matrix the lower matrices are transposed.

The optimal weight vector $\boldsymbol{w}$ is as given by equation 12.

The threshold $\boldsymbol{b} \in \mathcal{G}_n^D$ can be computed by using KKT conditions with the Clifford support vectors as follows

$$\begin{aligned} \boldsymbol{b} &= \Big[\boldsymbol{b}_1 \boldsymbol{b}_2 \boldsymbol{b}_3 \ldots \boldsymbol{b}_D \Big] \\ &= \Big[(b_{1s} + b_{1\sigma_1}\sigma_1 + \ldots + b_{1\sigma_1\sigma_2}\sigma_1\sigma_2 + \ldots + b_{1I} I) \\ &\qquad (b_{2s} + b_{2\sigma_1}\sigma_1 + \ldots + b_{2\sigma_1\sigma_2}\sigma_1\sigma_2 + \ldots + b_{2I} I) \ldots \\ &\qquad\qquad (b_{Ds} + b_{D\sigma_1}\sigma_1 + \ldots + b_{D\sigma_1\sigma_2}\sigma_1\sigma_2 + \ldots + b_{DI} I) \\ &= \sum_{j=1}^{l} (\boldsymbol{y}_j - \boldsymbol{w}^{*T} \boldsymbol{x}_j)/l. \end{aligned} \tag{26}$$

The decision function can be seen as sectors reserved for each involved class, i.e. in the case of complex numbers ($\mathcal{G}_{1,0,0}$) or quaternions ($\mathcal{G}_{0,2,0}$) we can see that the circle or the sphere are divide by means spherical vectors. Thus the decision function can be envisaged as

$$\begin{aligned} \boldsymbol{y} &= csign_m \Big[f(\boldsymbol{x}) \Big] = csign_m \Big[\boldsymbol{w}^{*T} \boldsymbol{x} + \boldsymbol{b} \Big] \\ &= csign_m \Big[\sum_{j=1}^{l} (\alpha_j \circ \boldsymbol{y}_j)(\boldsymbol{x}_j^{*T} \boldsymbol{x}) + \boldsymbol{b} \Big], \end{aligned} \tag{27}$$

where m stands for the state valency, e.g. bivalent, tetravalent and the operation "$\circ$" is defined as

$$\begin{aligned} (\alpha_j \circ \boldsymbol{y}_j) =& < \alpha_j >_0 < \boldsymbol{y}_j >_0 + < \alpha_j >_1 < \boldsymbol{y}_j >_1 \sigma_1 + \\ & \ldots + < \alpha_j >_{2^n} < \boldsymbol{y}_j >_{2^n} I, \end{aligned} \tag{28}$$

simply one consider as coefficients of the multivector basis the multiplications between the coefficients of blades of same degree. For clarity we introduce this operation "$\circ$"which takes place implicitly in previous equation (16).

Note that the cases of 2-state and 4-state (Complex numbers) can be solved by the multi-class real valued SVM, however in case of higher representations like the 16-state using quaternions, it would be awkward to resort to the multi-class real valued SVMs.

The major advantage of this approach is that one requires only one CSVM which even can admit multiple multivector inputs. A naive and time consuming approach will be to use a a set of real valued SVM.

4.2 Nonlinear Clifford Valued Support Vector Machines for Classification

For the nonlinear Clifford valued classification problems we require a Clifford valued kernel k($\boldsymbol{x}$,$\boldsymbol{y}$). In order to fulfill the Mercer theorem we resort to a component-wise Clifford-valued mapping

$$\begin{aligned}\boldsymbol{x} \in G_n \xrightarrow{\phi} \Phi(\boldsymbol{x}) = \Phi_s(x) + \Phi_{\sigma_1}\sigma_1 + \\ ... + \Phi_{\sigma_1}\sigma_2(x)\sigma_2 + ... + I\Phi_I(x) \in G_n.\end{aligned} \tag{29}$$

In general we build a Clifford kernel $k(x_m, x_j)$ by taking the Clifford product between the conjugated of $\boldsymbol{x}_m$ and $\boldsymbol{x}_j$ as follows

$$k(x_m, x_j) = \Phi(\boldsymbol{x})^*\Phi(\boldsymbol{x}), \tag{30}$$

note that the kind of conjugation operation $()^*$ of a multivector depends of the signature of the involved geometric algebra $\mathcal{G}_{p,q,r}$.

Quaternion-valued Gaussian window Gabor kernel function (we use here $\boldsymbol{i} = \sigma_2\sigma_3$, $\boldsymbol{j} = -\sigma_3\sigma_1$, $\boldsymbol{k} = \sigma_1\sigma_2$):

The Gaussian window Gabor kernel function reads

$$k(\boldsymbol{x}_m, \boldsymbol{x}_n) = g(\boldsymbol{x}_m, \boldsymbol{x}_n)exp^{-\boldsymbol{i}\mathbf{w}_0^T(\boldsymbol{x}_m - \boldsymbol{x}_n)} \tag{31}$$

where the normalized Gaussian window function is given by

$$g(\boldsymbol{x}_m, \boldsymbol{x}_n) = \frac{1}{\sqrt{2\pi}\rho}exp^{\frac{||\boldsymbol{x}_m - \boldsymbol{x}_n||^2}{2\pi^2}} \tag{32}$$

and the variables $\boldsymbol{w}_0$ and $\boldsymbol{x}_m - \boldsymbol{x}_n$ stand for the frequency and space domains respectively.

As opposite as the Hartley transform or the 2D complex Fourier this kernel function separates nicely the even and odd components of the involved signal, i.e.

$$\begin{aligned}k(\boldsymbol{x}_m, \boldsymbol{x}_n) &= k(\boldsymbol{x}_m, \boldsymbol{x}_n)_s + k(\boldsymbol{x}_m, \boldsymbol{x}_n)_{\sigma_2\sigma_3} + k(\boldsymbol{x}_m, \boldsymbol{x}_n)_{\sigma_3\sigma_1} + k(\boldsymbol{x}_m, \boldsymbol{x}_n)_{\sigma_1\sigma_2} \\ &= g(\boldsymbol{x}_m, \boldsymbol{x}_n)cos(\mathbf{w}_0^T\boldsymbol{x}_m)cos(\mathbf{w}_0^T\boldsymbol{x}_m) + g(\boldsymbol{x}_m, \boldsymbol{x}_n)cos(\mathbf{w}_0^T\boldsymbol{x}_m)sin(\mathbf{w}_0^T\boldsymbol{x}_m)\boldsymbol{i} \\ &+g(\boldsymbol{x}_m, \boldsymbol{x}_n)sin(\mathbf{w}_0^T\boldsymbol{x}_m)cos(\mathbf{w}_0^T\boldsymbol{x}_m)\boldsymbol{j} + g(\boldsymbol{x}_m, \boldsymbol{x}_n)sin(\mathbf{w}_0^T\boldsymbol{x}_m)sin(\mathbf{w}_0^T\boldsymbol{x}_m)\boldsymbol{k}.\end{aligned}$$

Since $g(\boldsymbol{x}_m, \boldsymbol{x}_n)$ fulfills the Mercer's condition it is straightforward to prove that $k(\boldsymbol{x}_m, \boldsymbol{x}_n)_u$ in the above equations satisfy these conditions as well.

After we defined these kernels we can proceed in the formulation of the SVM conditions. We substitute the mapped data $\Phi(\boldsymbol{x}) = \sum_{u=1}^{2^n} < \Phi(\boldsymbol{x}) >_u$ into the linear function $f(\boldsymbol{x}) = \boldsymbol{w}^{*T}\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{w}^{*T}\Phi(\boldsymbol{x}) + \boldsymbol{b}$. The problem can be stated similarly as equations (15-17). In fact we can replace the kernel function in equations 24 to accomplish the Wolfe dual programming and thereby to obtain the kernel function group for nonlinear classification

$$\begin{aligned} \boldsymbol{H}_s &= [k_s(\boldsymbol{x}_m, \boldsymbol{x}_j)]_{m,j=1,..,l} \\ \boldsymbol{H}_{\sigma_1} &= [k_{\sigma_1}(\boldsymbol{x}_m, \boldsymbol{x}_j)]_{m,j=1,..,l} \\ &\cdots \\ \boldsymbol{H}_{\sigma_n} &= [k_{\sigma_n}(\boldsymbol{x}_m, \boldsymbol{x}_j)]_{m,j=1,..,l}\,. \\ &\cdot \\ H_I &= [k_I(\boldsymbol{x}_m, \boldsymbol{x}_j)]_{m,j=1,..,l}\,. \end{aligned} \tag{33}$$

In the same way we use the kernel functions to replace the the dot product of the input data in the equation (27). In general the output function of the nonlinear Clifford SVM reads

$$\begin{aligned} \boldsymbol{y} = csign_m\Big[f(\boldsymbol{x})\Big] &= csign_m\Big[\boldsymbol{w}^{*T}\Phi(\boldsymbol{x}) + \boldsymbol{b}\Big] \\ &= csign_m\Big[\sum_{j=1}^{l}(\alpha_j \circ \boldsymbol{y}_j)(k(\boldsymbol{x}_j, \boldsymbol{x}) + \boldsymbol{b}\Big]. \end{aligned} \tag{34}$$

where m stands for the state valency.

Next we present the well known 2-D spiral problem to the 3-D space. This experiment should test whether the CSVM would be able to separate three 1-D manifolds embedded in $\mathbb{R}^3$. In Figure 4 on can see that the problem is nonlinear

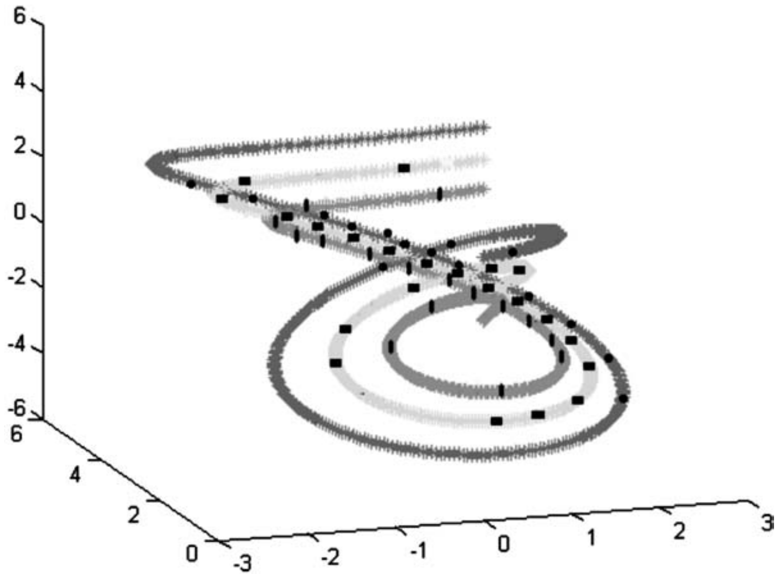

Fig. 4. 3D spiral with three classes. The marks represent the support vectors found by the CSVM.

separable. The CSVM used the kernel given by the equation 33. The CSVM found 16 support vector for $f_1(t)$, 21 support vector for $f_2(t)$ (in the middle) and 16 support vector for $f_3(t)$. Note that the CSVM indeed manage to separate the three classes. If we think in a real valued SVM (naive approach), one will require to do the job three SVMs.

5 Conclusions

In this article we have chosen the coordinate-free system of Clifford or geometric algebra for the analysis and design of algorithms useful for perception, action and learning.

Future intelligent machines will necessarily require of a powerful geometric language for reasoning at high level. The author believes that Clifford geometric algebra is a promissory mathematical system for representing and processing complex geometric data in real time.

Acknowledgment

I am very thankful to my PhD students: Leo Reyes Lozano, Julio Zamora Esquivel and Nancy Arana Daniel who provided me with experimental results useful for illustrating the application of geometric algebra in robotic vision and learning. Eduardo Bayro Corrochano, was supported by the project 49 of CONACYT-Fondo Sectorial de Investigación en Salud y Seguridad Social.

References

1. Bayro-Corrochano E. 2001. *Geometric Computing for Perception Action Systems*, Springer Verlag, Boston.
2. Bayro-Corrochano E. Conformal geometric algebra for robot perception. To appear in the *Hanbook on Computational Geometry for Patter Recognition, Computer Vision, Neuralcomputing and Robotics*, Eduardo Bayro-Corrochano (ed.), Chap. 11, Springer Verlag, Heidelberg, 2004.
3. H. Li, D. Hestenes and A. Rockwood. Generalized homogeneous coordinates for computational geometry. In Geometric Computing with Clifford Algebra, G. Sommer (Ed.), Springer-Verlag, pp. 27-59, 2001.
4. V. Vapnik. Statistical Learning Theory. Wiley, New York, 1998.