Adv. Appl. Clifford Algebras 24 (2014), 533–552

0188-7009/020533-20
published online February 15, 2014
DOI 10.1007/s00006-014-0448-2

Advances in Applied Clifford Algebras

Robust Pose Control of Robot Manipulators Using Conformal Geometric Algebra

L. González-Jiménez*, O. Carbajal-Espinosa, A. Loukianov, and E. Bayro-Corrochano

Abstract. A controller, based on sliding mode control, is proposed for the n-link robotic manipulator pose tracking problem. The point pair (a geometric entity expressed in geometric algebra) is used to represent position and orientation of the end-effector of a manipulator. This permits us to express the direct and differential kinematics of the end-effector of the manipulator in a simple and compact way. For the control, a sliding mode controller is designed with the following properties: robustness against perturbations and parameter variations, finite time convergence, and easy implementation. Finally, the application, of the proposed controller in a 6 DOF robotic manipulator is presented via simulation.

Keywords. Serial manipulators, pose control, motors, conformal geometric algebra, sliding modes.

1. Introduction

Accuracy, precision and repeatability are necessary features to ensure a good performance of industrial robotic manipulators. Because the robotic manipulators are complex and highly nonlinear plants, usually accompanied with perturbations and parameter variations, a robust nonlinear controller is required to obtain a desired response of the robotic manipulator. In this work, we obtain the differential kinematic model for manipulators with n-DOF using Conformal Geometric Algebra (CGA) and propose a controller based on sliding modes [1] to obtain robustness and finite time stabilization of the error variables of the system. Among robust control methodologies for robotic manipulators, sliding mode control [1, 2] is one of the most effective

*Corresponding author.
This work was supported in part by Project SEP-CONACYT "Métodos Geométricos y Cognitivos para la percepción, aprendizaje, control y acción de humanoides" under Grant 82084 and PhD scholarships no. 219316 and no. 28824.

approaches because of its robustness to perturbations and parameter variations. Also, sliding mode control is obtained from a simple procedure, which impacts in a low computational cost in a real implementation.

The CGA allows the representation of rigid transformations (rotations, translations, screw motions and others) and geometric entities (points, lines, planes, circles, spheres, point pairs, etc) in a simple way. Moreover, the composition of several rigid transformations acting over a geometric entity can be computed as a sequence of geometric products of consecutive motors (the conformal entity that represents a 3D rigid transformation). This efficient representation will be exploited to obtain the direct and differential kinematics of robotic manipulators [3, 4, 5, 6, 7].

Using the CGA framework, a sliding mode controller is designed for the pose tracking problem for a n-manipulator, where the pose of the manipulator is represented using a single geometric entity: the point pair. The rest of the work is organized as follows. Section II presents an introduction to the Conformal Geometric Algebra. The kinematic model for the pose of robotic manipulators is obtained in section III. The design of the error variables and sliding mode controller in CGA are defined in section IV. Section V presents the application of the designed controllers in a robotic manipulator of 6-DOF, via simulation. Finally, some conclusions are given in section VI.

2. Geometric Algebra

Let G_n denote the geometric algebra of n dimensions, which is a graded linear space. As well as vector addition and scalar multiplication, G_n has a non-commutative product that is associative and distributive over addition. This is called the *geometric* or *Clifford product*.

The inner product of two vectors is the standard *scalar* or *dot* product, which produces a scalar. The outer or wedge product of two vectors is a new quantity, which we call a *bivector*. We think of a bivector as an oriented area in the plane containing the vectors a and b, that is formed by sweeping a along b. Thus, $b \wedge a$ will have the opposite orientation, making the wedge product anticommutative. The wedge product is immediately generalizable to higher dimensions. For example, $(a \wedge b) \wedge c$, a *trivector*, is interpreted as the oriented volume formed by sweeping the area $a \wedge b$ along vector c. The wedge product of k vectors is a k-blade, and such a quantity is said to have *grade* k. A *multivector* (the linear combination of objects of different grades) is a *homogeneous* k-vector if it contains terms of only a single grade k.

In this paper we will specify the geometric algebra G_n of the n dimensional space by $G_{p,q,r}$, where p, q, and r stand for the number of basis vectors that square to 1, -1, and 0, respectively, and fulfill $n = p + q + r$. We will use e_i to denote the $i - th$ basis vector, where $1 \leq i \leq n$. In geometric algebra,

$G_{p,q,r}$, the geometric product of two basis vectors, is defined as

$$e_i e_j = \begin{cases} 1 & for \quad i = j \in 1, \ldots, p \\ -1 & for \quad i = j \in p+1, \ldots, p+q \\ 0 & for \quad i = j \in p+q+1, \ldots, p+q+r \\ e_i \wedge e_j & for \quad i \neq j. \end{cases}$$

This leads to a basis for the entire algebra

$$\{1\}, \{e_i\}, \{e_i \wedge e_j\}, \{e_i \wedge e_j \wedge e_k\}, \ldots, \{e_1 \wedge e_2 \wedge \ldots \wedge e_n\}. \tag{2.1}$$

Any multivector can be expressed in terms of this basis. The multivectors can be of grade 0 (scalars), grade 1 (vectors), grade 2 (bivectors), grade 3 (trivectors), etc., up to grade n (n-vectors).

Any pair of multivectors can be multiplied using the geometric product. Consider two k-vectors A_r and B_s of grades r and s, respectively. The geometric product of these multivectors can be written as

$$A_r B_s = \langle AB \rangle_{r+s} + \langle AB \rangle_{r+s-2} + ... + \langle AB \rangle_{|r-s|}, \tag{2.2}$$

where $\langle \rangle_t$ is used to denote the t-grade part of multivector, e.g. consider the geometric product of two vectors

$$ab = \langle ab \rangle_0 + \langle ab \rangle_2 = a \cdot b + a \wedge b.$$

2.1. Conformal Geometric Algebra

Geometric algebra $G_{4,1} = G_{4,1,0}$ can be used to treat conformal geometry in a very elegant way. To see how this is possible, we follow the same formulation presented in [6] and show how the Euclidean vector space $\mathbb{R}^3$ is represented in $\mathbb{R}^{4,1}$. This space has an orthonormal vector basis given by $\{e_i\}$ and $e_{ij} = e_i \wedge e_j$ are bivectorial bases and the bivector basis e_{23}, e_{31} and e_{12} that corresponds together with 1 to Hamilton's quaternions.

The unit Euclidean pseudo-scalar $I_e = e_1 \wedge e_2 \wedge e_3$, the bivector or Minkowski plane $E := e_4 \wedge e_5 = e_4 e_5$ and a pseudo-scalar $I = I_e E$ are used for computing Euclidean and conformal duals of multivectors. For more about conformal geometric algebra see [5, 6].

2.1.1. The Point. The vector $x_e \in \mathbb{R}^3$ representing a point after a conformal mapping is rewritten as

$$x_c \quad = \quad x_e + \frac{1}{2} x_e^2 e_\infty + e_0, \tag{2.3}$$

where the null vectors are the point at infinity $e_\infty = e_4 + e_5$ and the origin point $e_0 = \frac{1}{2}(e_4 - e_5)$, with the properties $e_\infty^2 = e_0^2 = 0$ and $e_\infty \cdot e_0 = 1$.

2.1.2. Lines. Lines can be defined in its dual form or OPNS (Outer Product Null Space), by the wedge product of two conformal points and the point at infinity as

$$L^* = x_{c_1} \wedge x_{c_2} \wedge e_\infty. \tag{2.4}$$

The standard IPNS (Inner Product Null Space) form of the line can be expressed as

$$L = \boldsymbol{n} I_e - e_\infty \boldsymbol{m} I_e, \tag{2.5}$$

where $\boldsymbol{n}$ and $\boldsymbol{m}$ stand for the line orientation and moment, respectively. The line in the IPNS standard form is a bivector representing the six Plücker coordinates.

2.1.3. Point Pair. The point pair is represented as

$$P_p^* = p_1 \wedge p_2 \tag{2.6}$$

and is obtained using the wedge product of the two points that define the point pair, p_1 and p_2. It can be obtained as the intersection of a line and a sphere, a line and a circle or a circle and a sphere. We can retrieve the points that compose the point pair using

$$p_{1,2} = \frac{P_p^* \pm \left(P_p^* \cdot P_p^*\right)^{1/2}}{-e_\infty \cdot P_p^*}. \tag{2.7}$$

These entities are useful to represent the parts of a robotic manipulator; for example, the line is used to express the joint axes of each D.O.F. of the robot and the point pair to model the end-effector of the manipulator.

2.2. Rigid Transformations

We can express rigid transformations in conformal geometry carrying out plane reflections.

2.2.1. Reflection. The combination of reflections of conformal geometric entities enables us to form other transformations. The reflection of a point x with respect to the plane π is equal to x minus twice the directed distance between the point and plane. That is, $ref(x) = x - 2(\pi \cdot x)\pi^{-1}$. We get this expression by using the reflection $ref(x_c) = -\pi x_c \pi^{-1}$ and the property of Clifford product of vectors $2(b \cdot a) = ab + ba$.

For an IPNS geometric entity Q, the reflection with respect to the plane π is given as

$$Q' \quad = \quad \pi Q \pi^{-1}. \tag{2.8}$$

2.2.2. Reversion. The reversion of an r-grade multivector $A_r = \sum_{i=0}^{r} \langle A_r \rangle_i$ is defined as:

$$\widetilde{A}_r = \sum_{i=0}^{r} (-1)^{\frac{i(i-1)}{2}} \langle A_r \rangle_i . \tag{2.9}$$

The reversion can also be obtained by reversing the order of basis vectors making up the blades in a multivector and then rearranging them in their original order using the anticommutativity of the Clifford product [5].

2.2.3. Translation. The translation of conformal geometric entities can be done by carrying out two reflections in parallel planes π_1 and π_2. That is,

$$Q' = \underbrace{(\pi_2\pi_1)}_{T_a} Q \underbrace{(\pi_1^{-1}\pi_2^{-1})}_{\widetilde{T_a}}, \tag{2.10}$$

$$T_a = (n + de_\infty)n = 1 + \frac{1}{2}ae_\infty = e^{\frac{a}{2}e_\infty}, \tag{2.11}$$

with $a = 2dn$, where n is the normal of both planes and d is the Hesse distance from the origin to the plane.

2.2.4. Rotation. The rotation is the product of two reflections at nonparallel planes that pass through the origin:

$$Q' = \underbrace{(\pi_2\pi_1)}_{R_\theta} Q \underbrace{(\pi_1^{-1}\pi_2^{-1})}_{\widetilde{R_\theta}}, \tag{2.12}$$

or by computing the Clifford product of the normals of the planes:

$$R_\theta = n_2 n_1 = \cos(\frac{\theta}{2}) - \sin(\frac{\theta}{2})l = e^{-\frac{\theta}{2}l}, \tag{2.13}$$

with $l = n_2 \wedge n_1$, and θ twice the angle between the planes π_2 and π_1.

The screw motion, called *motor*, related to an arbitrary axis L is

$$M_\theta = TR\widetilde{T} \tag{2.14}$$

and is applied in the same way as a rotor; that is,

$$Q' = \underbrace{(TR\widetilde{T})}_{M_\theta} Q \underbrace{(T\widetilde{R}\widetilde{T})}_{\widetilde{M_\theta}}, \tag{2.15}$$

$$M_\theta = TR\widetilde{T} = \cos(\frac{\theta}{2}) - \sin(\frac{\theta}{2})L = e^{-\frac{\theta}{2}L}, \tag{2.16}$$

where L is an arbitrary axis defined by a normalized line.

3. Kinematic Modeling Of Manipulators

The *direct kinematics* of a manipulator consists of calculating the position and orientation of the end-effector of a serial robot using the values of the joint variables. The joint variable is a translation $M_i = T_i = \exp^{-dne_\infty}$ for a prismatic joint and a rotation $M_i = R_i = \exp^{\frac{-\theta L_r}{2}}$ for a revolute joint, see Figure 1.

The direct kinematics for a serial robot is computed as a successive multiplication of motors given by

$$Q' = M_1 \cdots M_n Q \widetilde{M_n} \cdots \widetilde{M_1} = \left(\prod_{i=1}^{n} M_i Q \prod_{i=1}^{n} \widetilde{M}_{i=n-i+1} \right). \tag{3.1}$$

This equation is valid for points (i.e., the position of the end-effector), lines (i.e., the orientation of the end-effector), planes, circles, pair of points and spheres.

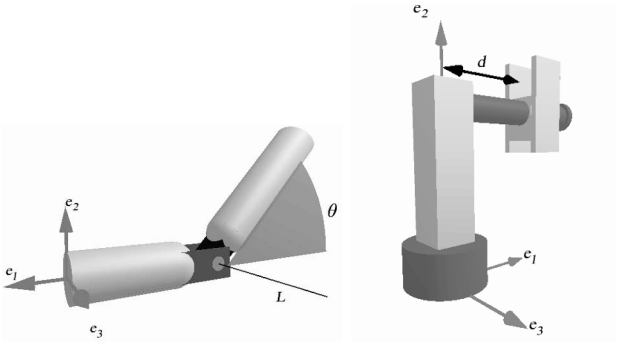

FIGURE 1. The revolute joint has an axis L and an angle θ of rotation (Left). The prismatic joint produces a displacement along the direction and magnitude of d (Right).

If we define the initial pose of the end-effector for an n-manipulator as the point pair $P_p^* = p_1 \wedge p_2$ where the orientation of the end-effector is defined from p_1 to p_2 and the current position of the end-effector is given by p_2. Then we can obtain its current pose P_p' using the direct kinematics of the manipulator defined as in (3.1) for a given angular position vector $q = [q_1 \ \dots \ q_n]^T$ and M_i as the motor for the i^{th} joint of the form

$$P_p' = M_1 \cdots M_n P_p \tilde{M}_n \cdots \tilde{M}_1. \tag{3.2}$$

The advantage of this approach is that with only one geometric entity it can be defined the pose of the manipulator, using a traditional method for modeling the pose of the robot, like matrices [11], it is needed to define an homogeneous transformation of the form

$$T_n^0 = \begin{pmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{pmatrix}, \tag{3.3}$$

where R_n^0 its a matrix that represent the orientation and o_n^0 is the position of the end-effector, respectively. Due to the number of entries of the matrix T_n^0 and the number of sums and multiplication necessary to use this matrix, it looks very attractive using a geometric entity of CGA which encloses all the information about the pose of the manipulator, to reduce the computational burden in real time implementations. Figure 2 shows a scheme of the proposed method to model the end-effector pose.

Differentiation of (3.2) yields

$$\dot{P}_p' = \sum_{j=1}^{n} \frac{\partial}{\partial q_j} \left[\left(\prod_{i=1}^{n} M_i \right) P_p \left(\prod_{i=1}^{n} M_{n-i+1} \right) \right] \dot{q}_j. \tag{3.4}$$

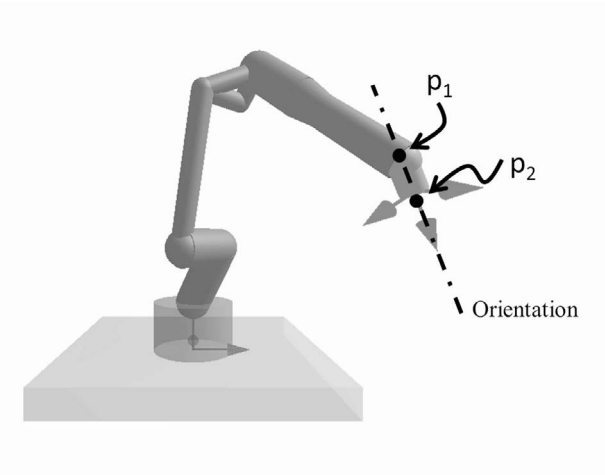

FIGURE 2. Point Pair modeling the pose of a n-link manipulator

The partial derivative can be expanded (as a product of two functions of q_k) resulting in

$$\dot{P}'_p = \sum_{j=1}^{n} \left(\frac{\partial}{\partial q_j} \left[\prod_{i=1}^{j} M_i \right] \Psi + \Omega \frac{\partial}{\partial q_k} \left[\prod_{i=n-j+1}^{n} \tilde{M}_{n-i+1} \right] \right) \dot{q}_j, \tag{3.5}$$

where $\Psi = \prod_{i=j+1}^{n} M_i P_p \prod_{i=1}^{n} \tilde{M}_{n-i+1}$ and $\Omega = \prod_{i=1}^{n} M_i P_p \prod_{i=1}^{n-j} \tilde{M}_{n-i+1}$.

A motor is defined as $M = e^{-qL/2}$, where q is the angle, then the differential of the motor is $\dot{M} = -\frac{1}{2} M L \dot{q}$, and for the reversion of a motor we get $\widetilde{\dot{M}} = \frac{1}{2} \dot{M} L \dot{q}$. Thus, we can rewrite the differential of the products of motors as

$$\begin{aligned} \tfrac{\partial}{\partial q_k} \left[\prod_{i=1}^{j} M_i \right] &= -\tfrac{1}{2} \prod_{i=1}^{j} M_i L_j = -\tfrac{1}{2} \left(\prod_{i=1}^{j-1} M_i \right) L_j M_j, \\ \tfrac{\partial}{\partial q_k} \left[\prod_{i=n-j+1}^{n} \tilde{M}_{n-i+1} \right] &= \tfrac{1}{2} \tilde{M}_j L_j \prod_{i=n-j+2}^{n} \tilde{M}_{n-i+1}. \end{aligned} \tag{3.6}$$

By substitution of (3.6) in (3.5) we obtain

$$\dot{P}'_p = \sum_{j=1}^{n} \left(-\frac{1}{2} \prod_{i=1}^{j-1} M_i L_j M_j \Psi + \frac{1}{2} \Omega \tilde{M}_j L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \dot{q}_j. \tag{3.7}$$

Now, using the following property

$$\prod_{i=1}^{j-1} \tilde{M}_{j-i} \prod_{i=1}^{j-1} M_i = 1. \tag{3.8}$$

Using $\Phi = \prod_{i=1}^{j-1} \tilde{M}_{j-i} \prod_{i=1}^{j-1} M_i$, equation (3.7) becomes

$$\dot{P}'_p = \sum_{j=1}^{n} \left(-\frac{1}{2} \prod_{i=1}^{j-1} M_i L_j \left[\Phi\right] \zeta + \frac{1}{2} \omega \left[\Phi\right] L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right) \dot{q}_j, \tag{3.9}$$

where $\zeta = \prod_{i=j}^{n} M_i P_p \prod_{i=1}^{n} \tilde{M}_{n-i+1}$ and $\omega = \prod_{i=1}^{n} M_i P_p \prod_{i=1}^{n-j+1} \tilde{M}_{n-i+1}$, and can be simplified as

$$\dot{P}'_p = \sum_{j=1}^{n} \left(-\frac{1}{2} \left[\prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right] P'_p + \frac{1}{2} P'_p \left[\prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \right] \right) \dot{q}_j, \tag{3.10}$$

where from direct kinematics, we know that $P'_p = \prod_{i=1}^{n} M_i P_p \prod_{i=1}^{n} \tilde{M}_{n-i+1}$ and

$$L'_j = \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i}, \tag{3.11}$$

where L_i is the axis for the i^{th} joint in the initial position and L'_j in the current position.

Substituting equations (3.11) in (3.10) results in

$$\dot{P}'_p = \frac{1}{2} \sum_{j=1}^{n} \left(P'_p L'_j - L'_j P'_p \right) \dot{q}_j. \tag{3.12}$$

If we define

$$\alpha_j = \frac{1}{2} \left(P'_p L'_j - L'_j P'_p \right) \tag{3.13}$$

equation (3.12) can be rewritten as

$$\dot{P}'_p = J \dot{q}, \tag{3.14}$$

where $\dot{q} = [\dot{q}_1 \ \ldots \ \dot{q}_n]$ are the angular velocities of each joint, and

$$J = [\alpha_1 \ \ldots \ \alpha n] \tag{3.15}$$

which is defined as the *differential kinematic* of the manipulator. Again, as in the direct kinematics, our method allows us to calculate the differential kinematic for the pose of the manipulator with respect to one geometric entity. It has been demonstrated in [9], that the calculus of the Jacobian matrix for differential kinematic using the geometric approach decrease the computational burden. In that paper they rename the jacobian matrix as V and it is possible to get this matrix in $O(Log_2(n))$ using parallel computing and n^2 threads. Using matrices, we need to calculate one jacobian matrix for the position J_v of the end-effector and one more for the orientation J_ω [11], which increase the computational burden. For sake of comparison, we will present the calculus of the jacobian matrix for the linear velocity of the end-effector for revolute joints.

The linear velocity of the end-effector is $\dot{o}_n^0$, which is the derivative of the position o_n^0 (3.3). By the chain rule for differentiation

$$\dot{o}_n^0 = \sum_{2}^{i=1} \frac{\partial o_n^0}{\partial q_i} \dot{q}_i. \tag{3.16}$$

Thus, the $i-th$ column of the jacobian matrix J_v, is given by

$$J_{v_i} = \frac{\partial o_n^0}{\partial q_i}. \tag{3.17}$$

Then, taking (3.3), o_n^0 is given by [11]:

$$o_n^0 = R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} + o_{i-1}^0. \tag{3.18}$$

Now, using (3.18) in (3.16), differentiation of o_n^0, $q_i = \theta_i$, gives

$$\begin{aligned}
\frac{\partial}{\partial \theta_i} o_n^0 &= \frac{\partial}{\partial \theta_i} \left[R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} \right] \\
&= \frac{\partial}{\partial \theta_i} R_i^0 o_n^i + R_{i-1}^0 \frac{\partial}{\partial \theta_i} o_i^{i-1} \\
&= \dot{\theta}_i S(z_{i-1}^0) R_i^0 o_n^i + \dot{\theta}_i S(z_{i-1}^0) R_{i-1}^0 o_i^{i-1} \\
&= \dot{\theta}_i S(z_{i-1}^0) \left[R_i^0 o_n^i + R_{i-1}^0 o_i^{i-1} \right] \\
&= \dot{\theta}_i S(z_{i-1}^0)(o_n^0 - o^{i-1}),
\end{aligned}$$

where z_{i-1} is the axis of rotation of the $i-th$ joint, and $S()$ is a $skew-matrix$ with the next property, for any vectors a and p belonging to $\mathbb{R}^3$,

$$S(a)p = a \times p,$$

where $a \times p$ is the cross product between the vectors a and p. For a more detailed development of the above equation and properties of the skew matrix, please refer to [11].

4. Sliding Mode Controller

In this section, the output tracking problem for the pose of the end-effector of a manipulator will be solved using the geometric algebra approach and the sliding mode control method.

Figure 3 shows a general scheme for our case of study, where the current pose, P_p', and the reference pose, $P_{p\,ref}$, for the end-effector of a serial manipulator are depicted. The control objective is to make the end-effector and reference poses equal by means of reconfiguring the structure of robot kinematics through the actuators (electrical motors, pistons, valves) of the joints.

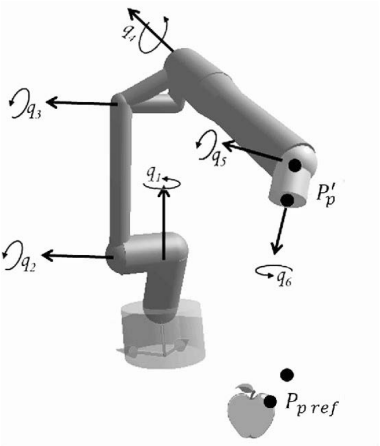

FIGURE 3. Current pose for the end-effector of a 6-DOF serial manipulator and the target $P_{p\,ref}$. Each q_i, $i = 1, \ldots, 6$ is and angular position

4.1. The pose tracking problem

Defining the following variables

$$x_1 = q, \quad x_2 = P'_p,$$

where $q = [q_1, q_2, \ldots, q_n]^T$ are the angular position of each joint and adding a disturbance term λ (due to external perturbations, model uncertainties and parameter variations), we can obtain a state-space model from (3.14) as

$$\begin{aligned} \dot{x}_1 &= u, \\ \dot{x}_2 &= Ju + \lambda, \end{aligned} \tag{4.1}$$

where $u = \dot{q}$ is the control term and the jacobian matrix J is defined as in (3.15).

Now, let $P_{p\,ref}$ be the reference for the pose of the end-effector expressed in conformal algebra, and propose the sliding surface for the controller as the tracking error variable of the form

$$s = P'_p - P_{p\,ref}, \tag{4.2}$$

which represents the difference between the reference pose and the current pose of the end-effector, this sliding surface is also a point pair as Figure 4 shows.

We assume that the disturbance term λ is bounded by positive scalar functions as

$$\|\lambda\| < \delta_1 |s| . \tag{4.3}$$

Then, the proposed controller is given by

$$u = -KJ^{+}\text{sign}(s), \tag{4.4}$$

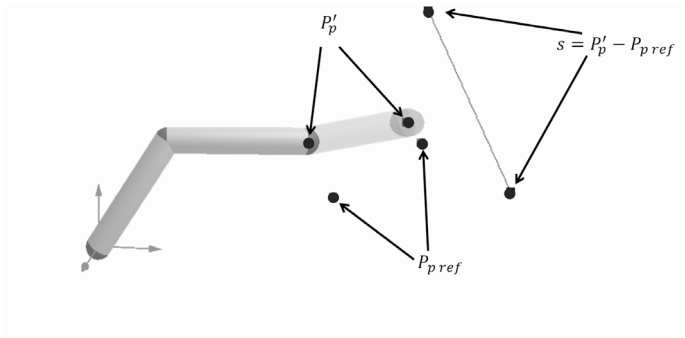

FIGURE 4. Geometric representation of the difference between $P_p^{'}$ and $P_{p\,ref}$.

where J^+ is the pseudo-inverse matrix of the jacobian matrix J and

$$K = diag\left\{ \ k_1, \quad \ldots, \quad k_6 \ \right\} \tag{4.5}$$

and $k_1, \ldots, k_6$ are scalars.

The closed loop dynamics for the sliding surface S is given by

$$\begin{aligned} \dot{s} &= \dot{P}_p{}' - \dot{P}_{p_{ref}} \\ &= Ju - \dot{P}_{p_{ref}} + \lambda. \end{aligned} \tag{4.6}$$

The stability conditions for (4.6) are given by

$$\|K\| > \left\| \dot{P}_{p_{ref}} \right\| + \|\lambda\|, \tag{4.7}$$

which are the standard stability conditions for sliding mode controllers. A detailed analysis for the aforementioned stability conditions can be found in [1, 12].

Due to the high frequency of the sliding mode controller, its implementation in real time becomes difficult. For this reason we will use the following definition:

Definition 4.1. The sign function can be approximated by the sigmoid function as shown by the following limit:

$$\lim_{\epsilon \to \infty} sigm\left(\epsilon, S\right) = sign\left(S\right). \tag{4.8}$$

Figure 5 shows the approximation for various of ϵ, which defines the slope of the sigmoid function for every S.

The sigmoid function that we used for this work is defined by

$$sigm\left(\epsilon, S\right) = \tanh\left(\epsilon S\right). \tag{4.9}$$

Now, we define the difference between the sign function and the sigmoid function as

$$\Delta\left(\epsilon, S\right) = sign\left(S\right) - sigm\left(\epsilon, S\right), \tag{4.10}$$

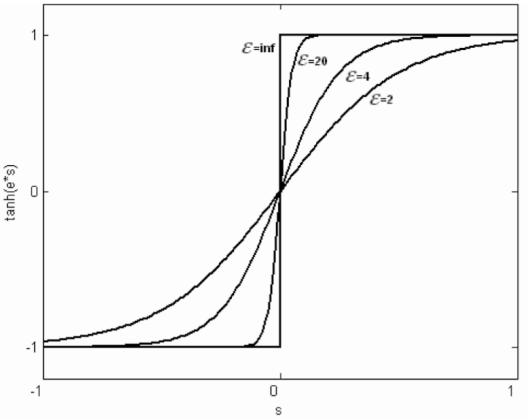

FIGURE 5. Approximation of the sign function using a sigmoid function for various values of parameter ϵ.

where $\Delta(\epsilon, S)$ is a bounded function by

$$\|\Delta(\epsilon, S)\|_2 \leq \xi, \tag{4.11}$$

where ξ is a positive constant scalar.

The $\Delta(\epsilon, S)$ value can be considered as a new disturbance and is added to the value of λ. The stability conditions for the controller using the sigmoid approximation are discussed by González et al in [10].

5. Simulations

The proposed control strategy was applied to the 6-DOF robotic manipulator shown in Figure 6. It is an industrial manipulator with 6 DOF that is located at the Automatic Control laboratory from the CINVESTAV, campus Guadalajara, Jalisco. This simulation was made using our own CGA libraries on Matlab [14] and CLUCalc [13].

The initial values for the rotation axes are given by

$$\begin{array}{l} L_{1,0} = e_{12},\ L_{2,0} = e_{31} + e_\infty(-0.15e_3 + 0.45e_1), \\ L_{3,0} = e_{13} + e_\infty(-1.02e_1 + 0.15e_3),\ L_{4,0} = e_{23} + e_\infty(-1.15e_2), \\ L_{5,0} = e_{13} + e_\infty(0.38e_3 - 1.15e_1),\ L_{6,0} = e_{12} + e_\infty(0.79e_2). \end{array} \tag{5.1}$$

The initial pose of the end-effector is given by

$$P_p^* = p_1 \wedge p_2, \tag{5.2}$$

where

$$p_1 = 0.79e_1 + 1.24e_3 + 1.09e_\infty + e_0,\ p_2 = 0.79e_1 + 2.24e_3 + 2.83e_\infty + e_0. \tag{5.3}$$

The point pair reference is defined as

$$P_{p\,ref} = p \wedge p_{ref},$$

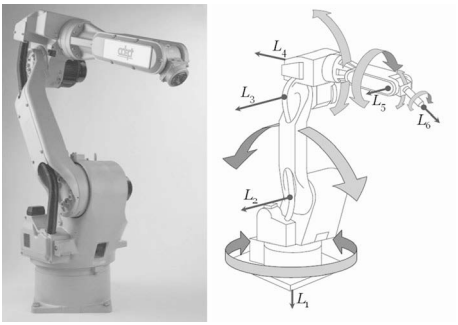

FIGURE 6. Industrial robotic manipulator Adept Six600 (left) and its 6DOF rotation axis (right).

where the Euclidean components for p and p_{ref} are given by

$$p_{ref} = \left[\begin{array}{cccc} \frac{6}{5}, & \frac{3}{10}\sin\left(3t+\frac{\pi}{2}\right), & \frac{3}{4}+\frac{3}{10}\sin\left(6t\right) \end{array} \right]^T,$$
$$p = \left[\begin{array}{cccc} \frac{2\sqrt{27}-5}{5\sqrt{3}}, & -\frac{1}{\sqrt{3}}+\frac{3}{10}\sin\left(3t+\frac{\pi}{2}\right), & \frac{\sqrt{27}-4}{\sqrt{48}}+\frac{3}{10}\sin\left(6t\right) \end{array} \right]^T.$$

The control parameters for (4.4) used in the simulation were selected manually

$$K = \operatorname{diag}\left\{ \begin{array}{cccccc} 0.8, & 0.8, & 0.8, & 5, & 6, & 6 \end{array} \right\}.$$

The six components for the disturbance term used in the simulation can be appreciated in Figure 7, we use the same perturbation for the joints 3 and 5 and also this is the case for the joints 4 and 6.

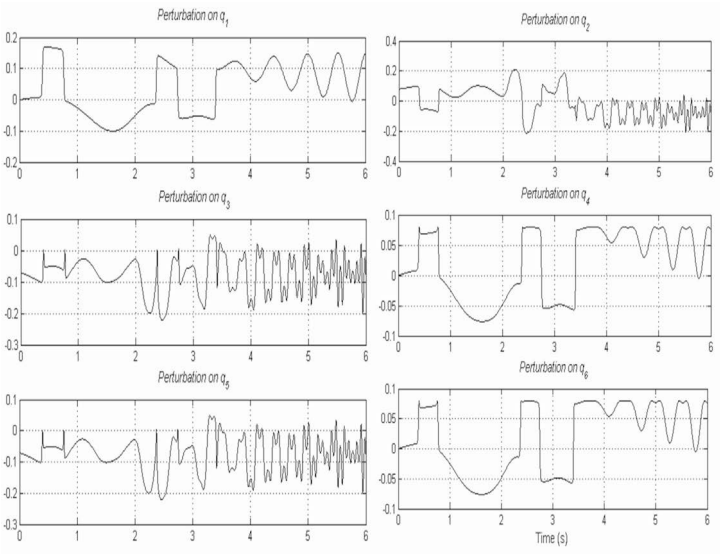

FIGURE 7. Six components of the disturbance term used in simulation.

The simulation results are presented in the following images. Figures 8 and 9 show the tracking response for the 3 Euclidean components for the orientation and position of the end-effector, respectively. It can be noted, for both cases, that the control objective is fulfilled.

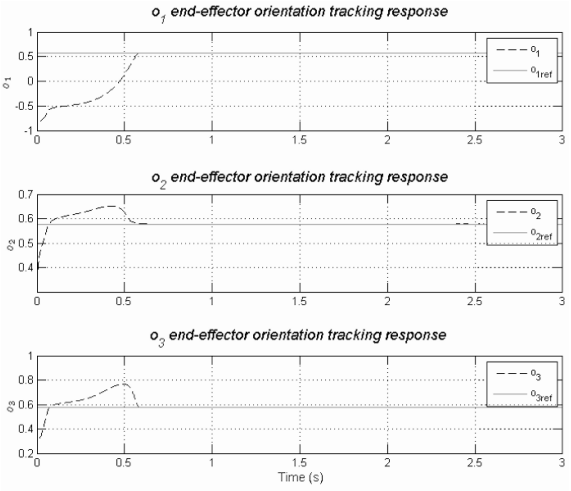

FIGURE 8. Tracking response for the orientation of the end-effector.

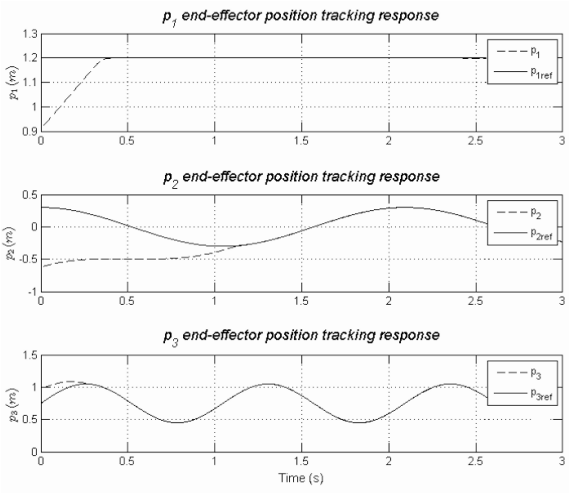

FIGURE 9. Tracking response for the position of the end-effector.

The orientation and position error variables can be seen in Figures 10 and 11, respectively. Their convergence to a vicinity of zero shows the robustness of the sliding mode control to external disturbances.

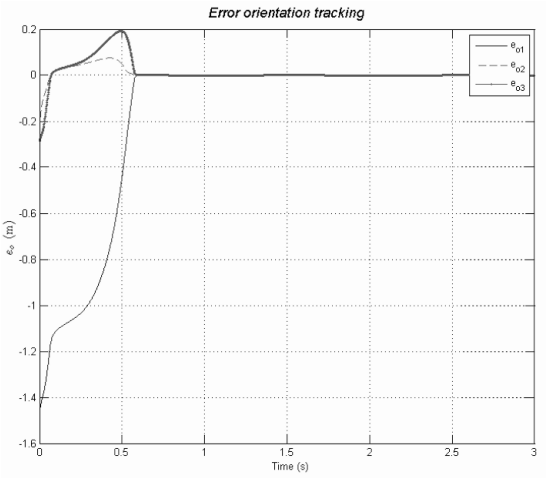

FIGURE 10. Components of the error variables for the orientation of the end-effector.

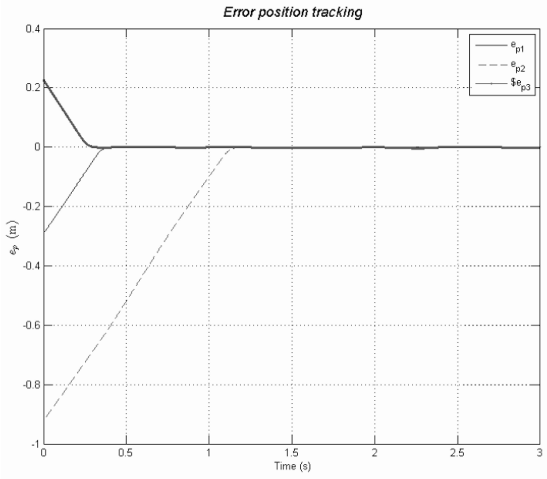

FIGURE 11. Components of the error variables for the position of the end-effector.

The angular positions for the 6 joints of the manipulator are shown in Figures 12 and 13.

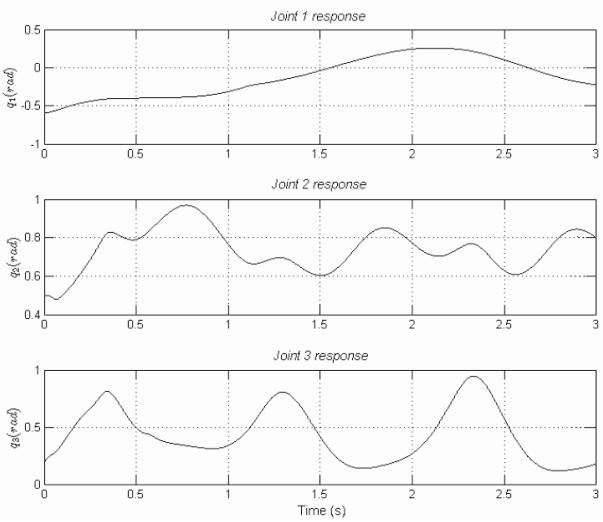

FIGURE 12. Angular positions for the joints 1, 2 and 3 of the manipulator.

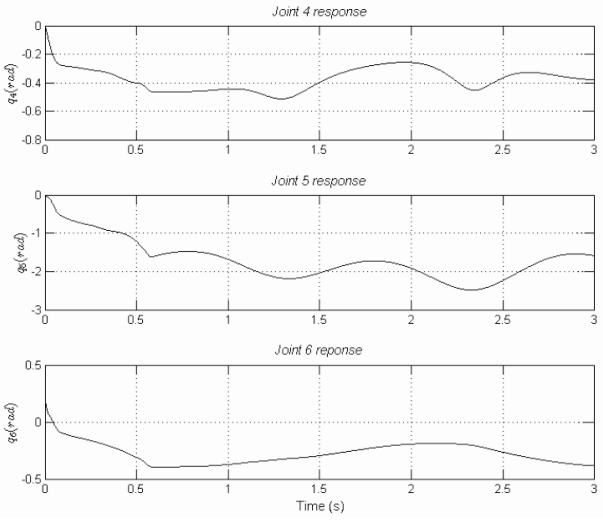

FIGURE 13. Angular positions for the joints 4, 5 and 6 of the manipulator.

Finally, the smooth control signals (angular velocities) can be seen in Figures 14 and 15. These are obtained by using a sigmoidal function.

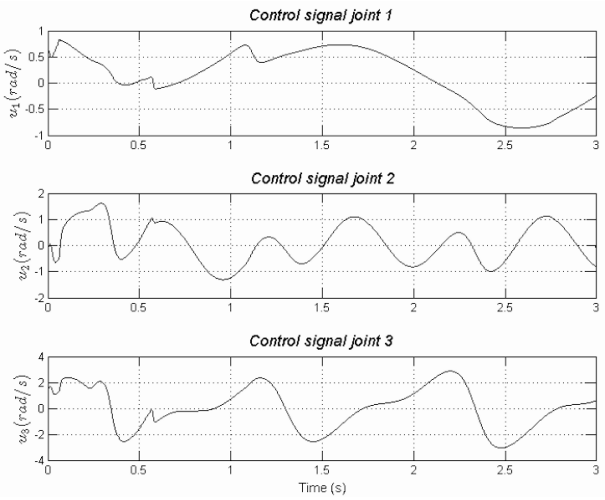

FIGURE 14. Control signals for the joint 1, 2 and 3 of the manipulator.

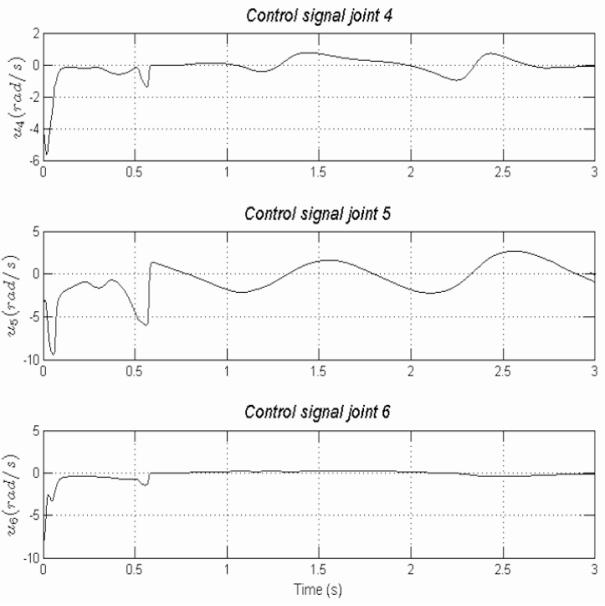

FIGURE 15. Control signals for the joint 4, 5 and 6 of the manipulator.

In order to compare our results against conventional approaches, we will show the simulation of the position tracking response of the end-effector and the control values using a matrices to model the pose of the manipulator and a standart sliding mode controller. The tracking response for position of the end-effector is shown in Figure 16.

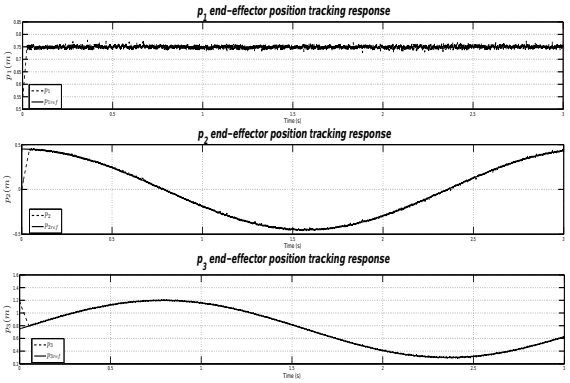

FIGURE 16. Tracking response for the position of the end-effector using matrices and sliding modes.

and the control laws are shown in the figure (17).

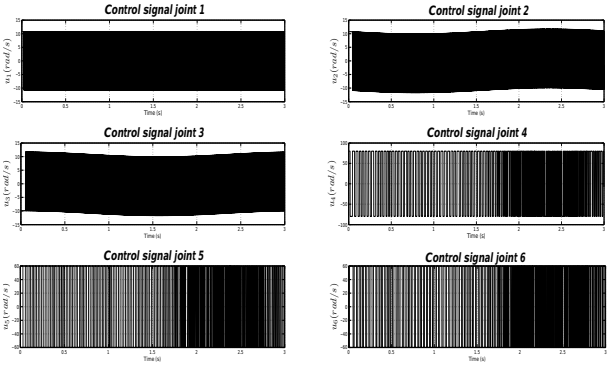

FIGURE 17. Control signals of the manipulator using matrices and sliding modes.

As you can see in Figure 16, the presence of chattering in the tracking is one issue when you use sliding modes control. Using the proposed control law, we get smooth control signals and good performance in the control of the pose of the manipulator.

6. Conclusions

A methodology for modeling and control of robotic manipulators was developed using the conformal geometric algebra framework. The pose for the manipulator was defined using a single geometric entity: the point pair. This is impossible using vector calculus, which shows the potential of the CGA. Moreover, a robust controller based in sliding mode control was designed for the tracking problem of the pose of the manipulator, and the simulation results showed the robustness of the proposed controller. Therefore, it can be concluded that CGA is a good framework for modeling and robust control of robotic manipulators.

References

[1] V. Utkin, J. Guldner and J. Shi, *Sliding Mode Control in Electromechanical Systems*. Taylor and Francis, 1999.

[2] L. González, A. Loukianov and E. Bayro-Corrochano, *Fully nested super-twisting algorithm for uncertain robotic manipulators*. IEEE International Conference on Robotics and Automation, 2011, 5807–5812.

[3] J. Zamora and E. Bayro, *Kinematics and Differential Kinematics of Binocular Heads*. IEEE International Conference of Robotics and Automation, 2006, 4130-4135.

[4] L. González, O. Carbajal-Espinosa and E. Bayro-Corrochano, *Geometric Techniques for the Kinematic Modeling and Control of Robotic Manipulators*. IEEE International Conference of Robotics and Automation, 2011, 5807–5812.

[5] Bayro-Corrochano Eduardo, *Geometric Computing for Wavelet Transforms, Robot Vision, Learning, Control and Action*. Springer Verlag, 2010.

[6] Li H., Hestenes DF. and Rockwood A., *Generalized Homogeneous coordinates for computational geometry*. Geometric Computing with Clifford Algebras, Springer-Verlag, 2001.

[7] L. Dorst, D. Fontijne and S. Mann, *Geometric Algebra for Computer Science*. Elsevier, 2007.

[8] C. Perwass, *Geometric Algebra with Applications in Engineering*. Springer-Verlag, 2009.

[9] Zamora J. and Bayro-Corrochano E., *Parallel forward Dynamics: A geometric approach*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, 2377–2382.

[10] L. González, A. Loukianov and E. Bayro-Corrochano, *Integral Nested Super-Twisting Algorithm for Robotic Manipulators*. IEEE International Conference on Intelligent Robots and Systems, 2010, 3580–3585.

[11] Mark W. Spong, Seth Hutchinson and M. Vidyasagar, *Robot Modeling and Control*. First Edition, John Wiley and Sons, 2005.

[12] H. Khalil, *Nonlinear Systems*. Prentice-Hall, 1996.

[13] CluCalc, *version 6.1.36*. Dr. Christian B. U. Perwass, 2010.

[14] Matlab, *version 7.6.0 (R2008a)*. The MathWorks Inc., 2008.

L. González-Jiménez
Instituto Tecnológico de Estudios Superiores de Occidente (ITESO A.C.)
Periférico Sur Manuel Gómez Morín 8585
C.P. 45604, Tlaquepaque, Jalisco
México
e-mail: luisgonzalez@iteso.mx

O. Carbajal-Espinosa, A. Loukianov and E. Bayro-Corrochano
CINVESTAV, Campus Guadalajara
Av. Del Bosque 1145, Col. El Bajío
C.P. 45019, Zapopan, Jalisco
México
e-mail: ocarbajal@gdl.cinvestav.mx
louk@gdl.cinvestav.mx
edb@gdl.cinvestav.mx

Received: December 31, 2012.
Accepted: December 02, 2013.