2014 14th IEEE-RAS International Conference on
Humanoid Robots (Humanoids)
November 18-20, 2014. Madrid, Spain

# Visual Servoing and Robust Object Manipulation Using Symmetries and Conformal Geometric Algebra

O. Carbajal-Espinosa[1], G. Osuna-González[2], L. González-Jiménez[3], A. Loukianov [4] and E. Bayro Corrochano[5]

***Abstract*— Object tracking and manipulation is an important process for many applications in robotics and computer vision. A novel 3D pose estimation of objects using reflectionally symmetry formulated in Conformal Geometric Algebra (CGA) is proposed in this work. The synthesis of the kinematics model for robots and a sliding mode controller using the CGA approach is described. Real time implementation results are presented for the pose estimation of object using a stereo vision system.**

## I. Introduction

Symmetry plays an important role in human activity since many objects found in domestic environments are symmetrical. The manner in which these objects are grasped and manipulated are also related to its axis of symmetry. For example, grasping of a bottle is done by applying force on opposite sides of its axis of symmetry, perpendicular to the axis. In this work, we obtain the pose of an reflectionally symmetric object and the differential kinematics model for manipulators with n-DOF using Conformal Geometric Algebra. Also, it is proposed a controller based on sliding modes [1] to obtain robustness and finite time stabilization of the error variables of the system for tracking and manipulation of objects.

The article is organized as follows. Section II presents an introduction to the Conformal Geometric Algebra. The kinematics model for the pose of robotic manipulators is obtained in section III. In section IV, the pose estimation algorithm is introduced and some real time results are presented. The design of the error variables and sliding mode controller in CGA are defined in section V. Section VI, presents the simulation results for a pose reference tracking, and the values of the controller gains. Finally, conclusions are given in section VII.

## II. 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*.

*This work was supported by Conacyt Ph.D. scholarship number 219316, and the master scholarship number 282138

[1,2,4,5]are with CINVESTAV, Department of Electrical Engineering and Computer Sciences, Campus Guadalajara ocarbajal[1],louk[3],edb[4]@gdl.cinvestav.mx.

[3]is with ITESO, Department of Electronic, Systems and Informatics, luisgonzalez@iteso.mx

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 $1 \leq i \leq n$.

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

$$A_r B_s = \langle AB \rangle_{r+s} + \langle AB \rangle_{r+s-2} + ... + \langle AB \rangle_{|r-s|}, \quad (1)$$

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

$$ab = \langle ab \rangle_0 + \langle ab \rangle_2 = a \cdot b + a \wedge b. \quad (2)$$

### A. 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 bivectors $e_{ij} = e_i \wedge e_j$. The bivector basis contains the bivectors $e_{23}$, $e_{31}$ and $e_{12}$ that corresponds 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].

*1) The point:* The vector $x_e \in \mathbb{R}^3$ representing a point after a conformal mapping is rewritten as

$$x_c = x_e + \frac{1}{2}x_e^2 e_\infty + e_0, \tag{3}$$

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) 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

$$L^* = x_{c_1} \wedge x_{c_2} \wedge e_\infty. \tag{4}$$

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

$$L = \boldsymbol{n} I_e - e_\infty \boldsymbol{m} I_e, \tag{5}$$

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.

*3) Pair of points:* The pair of points is represented as

$$P_p^* = p_1 \wedge p_2 \tag{6}$$

and is obtained using the wedge product of the two points that define the pair of points, $p_1$ 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 pair of points using

$$p_{1,2} = \frac{P_p^* \pm \left(P_p^* \cdot P_p^*\right)^{1/2}}{-e_\infty \cdot P_p^*}. \tag{7}$$

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 pair of points to model the end-effector of the manipulator.

### B. Rigid transformations

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

$$\widetilde{A}_r = \sum_{i=0}^{r} (-1)^{\frac{i(i-1)}{2}} \langle A_r \rangle_i . \tag{8}$$

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) Translation:* The translation of conformal geometric entities can be done by carrying out two reflections in parallel planes $\pi_1$ and $\pi_2$. That is,

$$Q' = \underbrace{(\pi_2\pi_1)}_{T_a} Q \underbrace{(\pi_1^{-1}\pi_2^{-1})}_{\widetilde{T_a}}, \tag{9}$$

$$T_a = (n + de_\infty)n = 1 + \frac{1}{2}ae_\infty = e^{\frac{a}{2}e_\infty}, \tag{10}$$

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

*3) Rotation:* The rotation is the product of two reflections at nonparallel planes that pass through the origin:

$$Q' = \underbrace{(\pi_2\pi_1)}_{R_\theta} Q \underbrace{(\pi_1^{-1}\pi_2^{-1})}_{\widetilde{R_\theta}}, \tag{11}$$

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

$$R_\theta = n_2 n_1 = cos(\frac{\theta}{2}) - sin(\frac{\theta}{2})l = e^{-\frac{\theta}{2}l}, \tag{12}$$

with $l = n_2 \wedge n_1$, and $\theta$ twice the angle between the planes $\pi_2$ and $\pi_1$.

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

$$M_\theta = TR\widetilde{T} \tag{13}$$

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

$$Q' = \underbrace{(TR\widetilde{T})}_{M_\theta} Q \underbrace{(T\widetilde{R}\widetilde{T})}_{\widetilde{M_\theta}}, \tag{14}$$

$$M_\theta = TR\widetilde{T} = cos(\frac{\theta}{2}) - sin(\frac{\theta}{2})L = e^{-\frac{\theta}{2}L}, \tag{15}$$

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

## III. Kinematics 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 $M_i = T_i = e^{-dne_\infty}$ for a prismatic joint and a rotation $M_i = R_i = e^{\frac{-\theta L_r}{2}}$ for a revolute joint.

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

$$Q' = M_1 \cdots M_n Q \widetilde{M}_n \cdots \widetilde{M}_1 = \left( \prod_{i=1}^{n} M_i Q \prod_{i=1}^{n} \widetilde{M}_{i=n-i+1} \right). \tag{16}$$

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.

The *differential kinematics* of the system results from the differentiation of (16) for points and lines, and is given by

$$\begin{aligned} \dot{x}'_p &= J_x \dot{q} \\ \dot{L}'_p &= J_L \dot{q} \end{aligned} \tag{17}$$

with $\dot{q} = [\dot{q}_1 \ \dots \ \dot{q}_n]$, and

$$J_x = \left[x'_p \cdot L'_1 \ \dots \ x'_p \cdot L'_n\right] \tag{18}$$

$$J_L = [\alpha_1 \ \dots \ \alpha n] \tag{19}$$

where

$$L'_j = \left( \prod_{i=1}^{j-1} M_i \right) L_j \left( \prod_{i=1}^{j-1} \widetilde{M}_{j-i} \right) \tag{20}$$

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

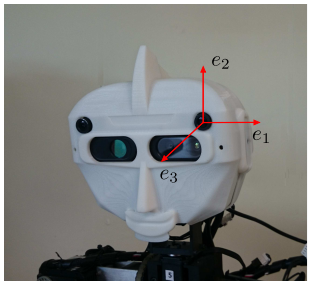
and $L_j$ is the axis for the $j^{th}$ joint in the initial position.

Equations (17) is defined as the *differential kinematics* of the manipulator. Again, as in the direct kinematics, our method allows us to calculate the differential kinematics 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 kinematics 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. Please refer to [3] for a more detailed explanation about the differentiation process.

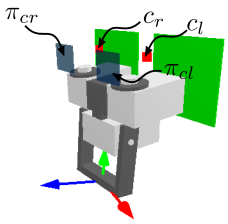
## IV. 3D Pose Estimation

Next we describe a procedure to obtain the 3D pose estimation for tracking and manipulation of the object. The perception system consists of a stereo vision system (SVS) mounted in a pan-tilt unit (PTU) as shown in figure 1(a). First, a calibration process for the SVS is realized. This process consists of retrieving the position and orientation of the principal axis of the right camera with respect to the principal axis of the left camera.

### A. Line symmetry estimation via reflectional symmetry




(a) SVS mounted in a PTU used to apply the proposed algorithm. The frame reference is fixed in left camera.




(b) Geometric entities in PTU

Fig. 1. Pan tilt unit

The procedure for pose estimation is based on a *Fast Reflectional Symmetry Detection* algorithm proposed in [15], [16]. In [17], the symmetry detection is combined with a block motion detector to find the symmetry axis of an object in motion, which is limited due to it is impossible to detect static objects. Instead of this, a color-based segmentation is proposed to solve this issue. The color segmentation is performed in HSV color space (hue-saturation-value); this color space is chosen due to its relative robustness to changes in the environment's illumination.

Once the image segmentation is obtained in each camera, it is converted to an edge image using an appropriate edge filter and after that the fast reflectional symmetry detection is applied, obtaining a parametrization of the line given by its angle and the radius or distance perpendicular to the line symmetry $(\theta, r)$ (see figure 2).

Using the data obtained from camera calibration (for both cameras), the line symmetry is transformed from the image coordinates to camera coordinates, i.e. from $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ and then create the line in conformal space as follows

$$L = \cos(\theta)e_{23} + \sin(\theta)e_{31} + re_3 \wedge e_\infty \tag{21}$$

Notice that from eq. (21), line $L$ lies in $xy$ plane. Since a rigid transformation $[R, t]$ relates both cameras, we need to define the line in both camera planes $\pi_{cl}$ and $\pi_{cr}$ (figure 1(b)). In order to obtain that, a motor $M$ is constructed with this transformation that relates the cameras. The lines are then defined as follows:

$$L_l = L \tag{22}$$
$$L_r = ML\widetilde{M} \tag{23}$$

where $L_l$ is the line of the left camera plane $\pi_{cl}$ and $L_r$ is from the right camera plane $\pi_{cr}$.

To obtain the 3D line symmetry, it is necessary to get the points at camera center which can be calculated from calibration matrix. Without loss of generality, it is possible to define the camera center of the left camera $c_l$ as the origin and the right camera center $c_r$ as a translated origin (defined by the rigid transformation M) i.e.

$$c_l = eo \tag{24}$$
$$c_r = Me_0\widetilde{M} \tag{25}$$

With the lines and the image center points obtained, we create two planes as follows

$$\pi_l^* = c_l \wedge L_l^* \tag{26}$$
$$\pi_r^* = c_r \wedge L_r^* \tag{27}$$

Finally, the symmetry line in 3D $L_{3D}$ is created by the intersection of the planes

$$L_{3D} = \pi_l \wedge \pi_r \tag{28}$$

The figure 3 shows a general scheme of the idea presented.

To create the reference for the object manipulation, (28) will be used as the orientation reference $L_{Mref}$. The subscript $M$ is used to define manipulation. In order to calculate a reference for the position, it is necessary to calculate a point that lies in the line symmetry of the object, which

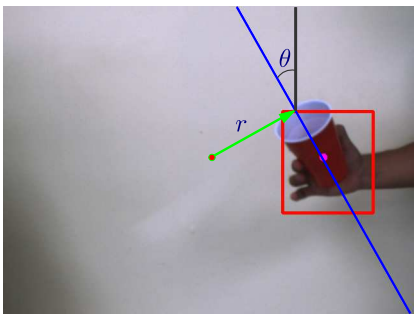
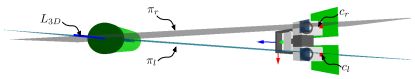
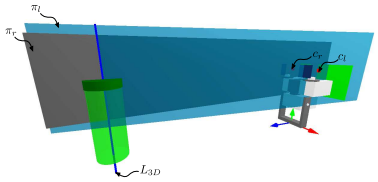



Fig. 2. Parameters obtained from fast reflectional symmetry detection



(a) Top view



(b) Isometric view

Fig. 3. Schematic representation of 3D line symmetry estimation

will be found calculating its mass center. In each segmented image the mass center is computed and the 3D point is calculated using the parameters obtained in calibration. Once this point is obtained, it is transformed to its conformal point representation $x_{Mref}$.

In order to create the visual tracking reference of the object, a line is defined using the mass center $x_{Mref}$ and left camera center $c_l$, as shown in the previous section (as in the manipulation reference the subscript $V$ will be used to define visual tracking).

$$L^*_{Vref} = x_{Mref} \wedge c_l \wedge e_\infty. \tag{29}$$

### B. Real time results

The results obtained of the real time implementation are presented in this section. The algorithm was developed using C++ and a computer vision library called IVT (Integrating Vision Toolkit) [18]. In order to obtain the line symmetry in the plane images, the line is projected in this plane as follows

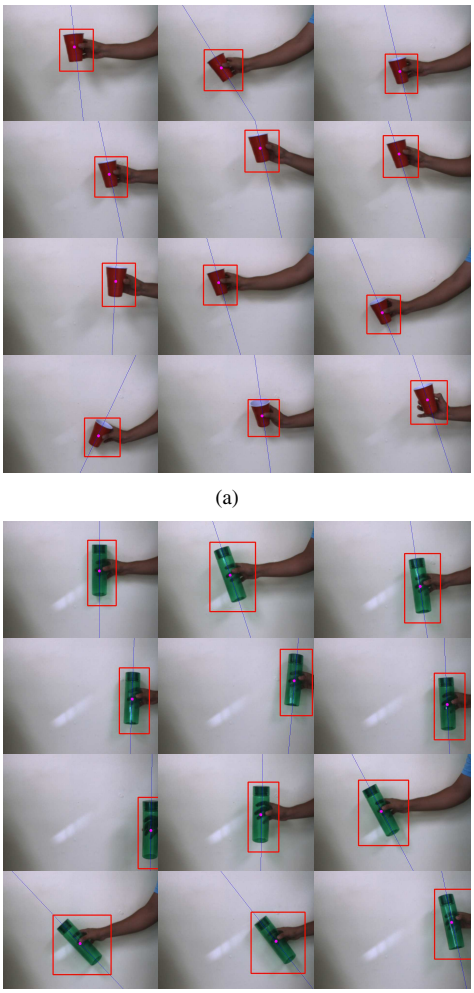
$$L_{i,l} = (L_{3D} \cdot \pi_{cl})\pi_{cl}^{-1} \tag{30}$$
$$L_{i,r} = (L_{3D} \cdot \pi_{cr})\pi_{cr}^{-1} \tag{31}$$

where $L_{i,l}$ and $L_{i,r}$ are the line symmetry in the images plane of the left and right camera respectively. The figure 4 shows the 3D line symmetry projected in the image plane obtained using different objects.

## V. 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.

The figure 5 shows a general scheme for our case of study. 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 of the joints.



(a)

(b)

Fig. 4. Sequence of left camera images from real time implementation

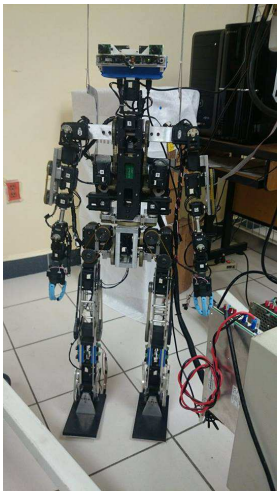


Fig. 5. MEXONE, humanoid robot of two 7-DOF robotic arm and a 2-DOF head, for real time implementation.

### A. The pose tracking problem

Defining the following variables

$$x'_p = pose$$

we can obtain a state-space model from (17) as

$$\dot{x}'_p = Ju + \lambda, \tag{32}$$

where $x'_p$ is the current pose of the end effector, $u = \dot{q}$ is the control term, the Jacobian matrix $J$ is defined as in (18) and adding a disturbance term $\lambda$ (due to external perturbations, model uncertainties and parameter variations).

Now, let $x_{ref}$ be the reference for the pose of the end-effector expressed in conformal algebra($x_{ref} = [x_{Mref}, L_{Mref}]^T$), and propose the sliding surface for the controller as the tracking error variable of the form

$$s = x'_p - x_{ref}. \tag{33}$$

We assume that the disturbance term $\lambda$ is bounded by positive scalar functions as

$$\|\lambda\| < \delta_1 |s|. \tag{34}$$

Then, the proposed controller is given by

$$u = -KJ^{+}\text{sign}(s), \tag{35}$$

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

$$K = diag\{\ k_1, \quad \ldots, \quad k_6\ \} \tag{36}$$

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

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

$$\begin{aligned} \dot{s} &= \dot{x}'_p - \dot{x}_{ref} \\ &= Ju - \dot{x}_{ref} + \lambda. \end{aligned} \tag{37}$$

The stability conditions for (37) are given by

$$\|K\| > \|\dot{x}_{ref}\| + \|\lambda\|, \tag{38}$$

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 5.1:* The sign function can be approximated by the sigmoid function as shown by the following limit:

$$\lim_{\epsilon \to \infty} sigm(\epsilon, S) = sign(S). \tag{39}$$

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

$$sigm(\epsilon, S) = \tanh(\epsilon S). \tag{40}$$

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

$$\Delta(\epsilon, S) = sign(S) - sigm(\epsilon, S), \tag{41}$$

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

$$\|\Delta(\epsilon, S)\|_2 \leq \xi, \tag{42}$$

where $\xi$ is a positive constant scalar.

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

Without loss of generality, the same control law can be implemented in the PTU for object tracking using $L_{Vref}$ as the reference.

## VI. Applied Controllers

Consider the system shown in figure 5, which is composed of a serial manipulator of 7-DOF. For a given target, the end-effector of the 7-DOF manipulator must realize pose tracking of the target. First, the kinematics model the manipulator is defined. Then, the parameters of the proposed controllers are determined. Finally, a simulation of the performance of the closed-loop system is presented.

The pose control term for the 7-DOF manipulator is defined as $u = \begin{bmatrix} \dot{q}_1 & \dot{q}_2 & \dot{q}_3 & \dot{q}_4 & \dot{q}_5 & \dot{q}_6 & \dot{q}_7 \end{bmatrix}^T$ and is obtained via equation (35). The control gain was selected as $K = diag\{7.5, \ldots, 7.5\}$. The reference vector used was

$$\begin{aligned} x_{ref} &= [-0.3 + 0.1\cos(2t), 0.15, 0.25, 0, 1, 0]^T, \\ \dot{x}_{ref} &= [-0.2\sin(2t), 0, 0, 0, 0, 0]^T \end{aligned} \tag{43}$$

### *Simulation Results*

The simulation process was developed in two steps. First, the differential kinematics model and the controllers for the robotic system were programmed in MatLab [14] using our own conformal geometric libraries, and the response for the closed loop system was obtained. Then, the data of the joint variables obtained from MatLab were used in a 3-D model of the robotic system developed in CLUCalc [13], in order to obtain a better visual appreciation of the behavior of the closed loop system.

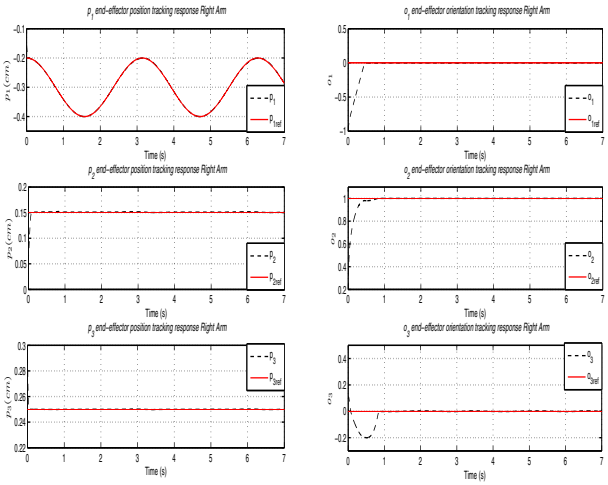


Fig. 6. Euclidean components for the position and orientation of the end-effector of the 7-DOF manipulator and their references.

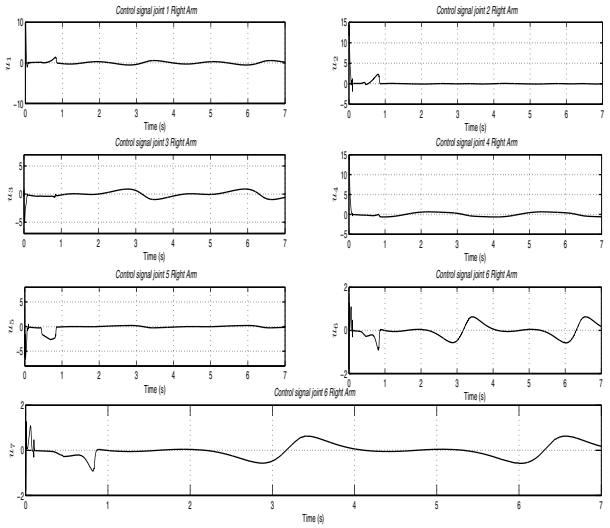



Fig. 7. Control values of joint velocities of the 7-DOF manipulator .

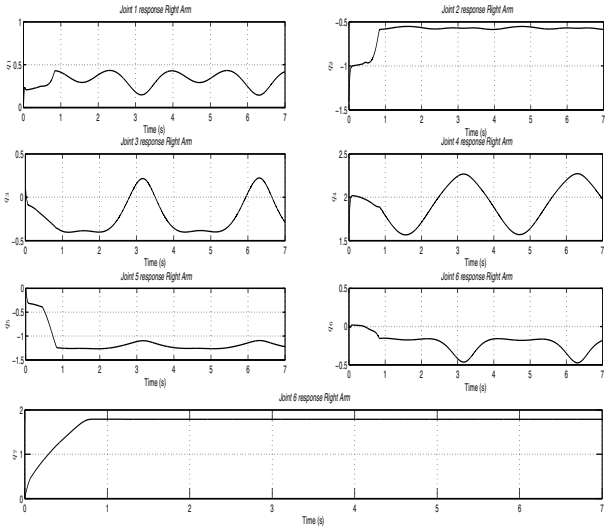



Fig. 8. Angular position of the joints of the 7-DOF manipulator .

The previous figures shows the simulation response of the pose tracking for the 7-DOF manipulator using the control laws proposed. The figure 6 shows the tracking response for the pose of the end-effector, the figure 7 depicts the control values obtained and finally the joint position of each joint are shown in the figure 8. It can be noted that the objective of control is fulfilled. The real time implementation is shown in the figure 9.

## VII. Conclusions

A methodology for object pose estimation via reflectional symmetry of objects, modeling and control of robotic manipulators was developed using the conformal geometric algebra framework. The pose for the manipulator was defined using lines and points. 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.

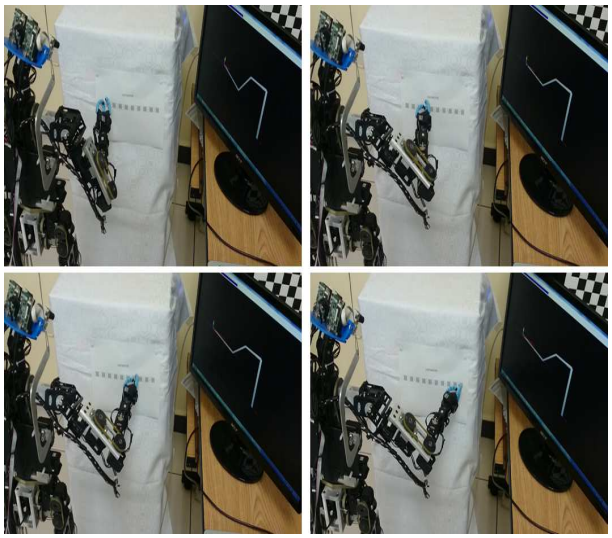


Fig. 9. Real time implementation of the proposed control law.

## References


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

[2] González, L.; Loukianov, A.; Bayro-Corrochano, E., *Fully nested super-twisting algorithm for uncertain robotic manipulators*, IEEE International Conference on Robotics and Automation (ICRA '11), Shanghai, China, 2011, pp. 5807-5812.

[3] Zamora, J.; Bayro-Corrochano, E., *Kinematics and Differential Kinematics of Binocular Heads*, IEEE International Conference of Robotics and Automation ICRA '06, Orlando, Florida, 2006, pp. 4130-4135.

[4] González, L.; Carbajal-Espinosa, O.; Bayro-Corrochano, E., *Geometric Techniques for the Kinematic Modeling and Control of Robotic Manipulators*, IEEE International Conference of Robotics and Automation ICRA '11, Shanghai, China, 2011, pp. 5807-5812.

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

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

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

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

[9] Zamora, J.; Bayro-Corrochano, E., *Parallel forward Dynamics: A geometric approach*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10), Taipei, Taiwan, 2010, pp. 2377-2382.

[10] González, L.; Loukianov, A.; Bayro-Corrochano, E., *Integral Nested Super-Twisting Algorithm for Robotic Manipulators*, IEEE International Conference on Intelligent Robots and Systems (IROS '10), Taipei, Taiwan, 2010, pp. 3580-3585.

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

[12] Khalil, H., *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.

[15] Li, W. H.; Zhang, A.; Kleeman, L., *Fast Global Reflectional Symmetry Detection for Robotic Grasping and Visual Tracking*. In Proceedings of Australasian Conference on Robotics and Automation ACRA '05, Sydney, Autralia, 2005.

[16] Wai Ho Li; Zhang, A.M.; Kleeman, L., *Real Time Detection and Segmentation of Reflectionally Symmetric Objects in Digital Images*. IEEE/RSJ International Conference on Intelligent Robots and Systems 2006, pp.4867,4873.

[17] Li, W. H.; Kleeman, L., *Real Time Object Tracking using Reflectional Symmetry and Motion*. Proc. IEEE International Conference on Intelligent Robots and Systems IROS '06, Beijing, China, 2006, pp. 2798-2803.

[18] http://ivt.sourceforge.net/ *The Integrating Vision Toolkit*