A Modular Neural Network linking Hyper RBF and AVITE models for reaching moving objects

J.L. Pedreño-Molina[1], J. Molina-Vilaplana[2], J. López-Coronado[2]

[1]*Department of Information Technologies and Communications*
[2]*Department of Systems Engineering and Automation*
Technical University of Cartagena. Campus Muralla del Mar,s/n. 30.202 Cartagena, SPAIN

Abstract. In this paper, the problem of precision reaching applications in robotic systems for scenarios with static and non-static objects has been considered and a solution based on a neural architecture biologically inspired has been proposed and implemented. The goal of this solution is to combine robustness and capability mapping trajectories from two biologically inspired neural networks: HypRBF and AVITE. The Hyper Basis Radial Function (HypRBF) neural model solves the inverse kinematic in redundant robotic systems, while the Adaptive Vector Integration to End-Point (AVITE) visuo-motor neural model quickly mapping the difference vector between current and desired position in both spatial (visual information) and motor coordinates (propioceptive information). The anthropomorphic behaviour of the proposed architecture for reaching and tracking tasks in presence of spatial perturbations has been validated over a real arm-head robotic platform.

***Keywords*:** Biological model, visuo-motor robotic system, reaching, tracking, regularization networks, learning inverse kinematics

1 Introduction

Most movement tasks in robotics are defined in coordinate systems that are different from the actuator space in which motor commands must be issued. Hence, movement planning and learning in task space require appropriated coordinate transformations from task to actuator space **[1,2,16]** before motor commands can be computed. The transformation from kinematic plans in external coordinates to internal robot coordinates is the classic inverse kinematics problem, a problem that arises from the fact that inverse transformations are often ill – posed. The approach taken in this paper is trying to solve this problem by using neural network learning of inverse kinematics of a redundant manipulator. Later we validate the model implementing it

on a real robot platform composed of an ABB industrial robot guided by an anthropomorphic vision stereohead.

Trajectories in task space must be carried out by articulator or end-effector movements. One possibility is to learn a position to position mapping from task space; e.g., each point in 3–D space can be mapped to a joint configuration that is satisfactory for this point. Another possibility is to use a directional mapping from desired movement direction in task space into movement directions in effector space (e.g., joint rotations). The model proposed in this paper uses the latter form of mapping through a neural network training, because it provides an automatic compensation for externally imposed constraints on effector motion. The use of a directional mapping for movement control is closely related to robotic controllers that utilize a generalized inverse of the Jacobian matrix **[3,8,11,12,14,17]** The relationship between spatial velocity of the end-effector and the joint velocities of a manipulator such as a robotic arm is given by the following equation:

$$\Delta x = J(\theta)\Delta\theta \qquad (1)$$

where Δx is the spatial velocity vector of the hand, $\Delta\theta$ is the joint velocity vector, and $J(\theta)$ is the manipulator's Jacobian matrix, whose elements depends only on the joint configuration θ. To obtain a joint velocity vector that moves the hand at a desired spatial velocity, we can rearrange this equation

$$\Delta\theta = J^{-1}(\theta)\Delta x \qquad (2)$$

where $J^{-1}(\theta)$ is an inverse of the Jacobian matrix. For a redundant manipulator, a unique inverse for J does nor exist. In this case, J^{-1} is a generalized inverse, or pseudoinverse, of the Jacobian matrix. The most commonly used generalized inverse is the Moore – Penrose pseudoinverse, which has the desiderable property of returning the minimum norm joint rotation vector that can produce the desired spatial velocity.

Learning inverse kinematics is useful when the kinematic model of the robot is not accurately available, when Cartesian information is provided in uncalibrated camera coordinates, or when the computational complexity of the solutions becomes too high. Learning methods are inherently self calibrating, avoiding problems due to kinematic singularities.

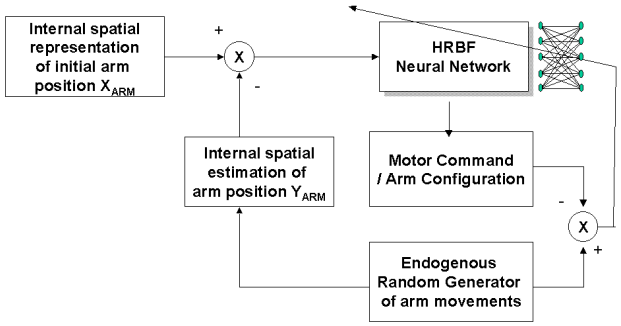


Figure 1. Learning scheme for acquisition of the directional mapping that allows the transformation between desired spatial or task coordinates increments or velocities into joint rotations of the robot manipulator.

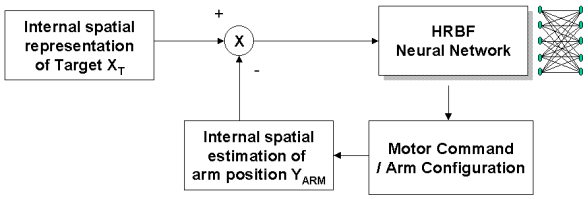


Figure 2.- During performance, the adaptive control scheme proposed in this paper continuously computes through vision the difference vector in task coordinates between actual position of the end-effector and target position. This difference vector and propioceptive information about the current joint configuration are used by the neural network to compute the joint rotations that allows the system to reach the target.

The learning paradigm used in this paper is the so called "direct inverse" (**[10]**; Figure 1). In this learning paradigm, movement commands are generated in effector space (typically random during training), and the system learns a mapping from the task space consequences of these movements to the movement commands that caused them. This inverse mapping can later be used to command effector space movements to achieve task space goals (Figure 2). Learning occurs during action perception cycles in which correct robot configurations are reached and the visual information associated with the end-effector displacement is correlated with the joint increments that allowed that displacement. So, inverse kinematic learning in that way will not demand impossible postures as can result from an ill conditioned matrix inversion.

The major obstacle in learning inverse kinematics lies in the problem that the inverse kinematics of a redundant kinematic chain has infinite solutions. Thus the

learning algorithm has to acquire a particular and a valid inverse kinematic solution. This issue was characterized by Jordan **[9]** Jordan and Rumelhart **[10]** as the problem of non – convex mappings. These authors shown that in order to learn the inverse kinematics it is necessary that all joints velocities generated during training form a convex set. Unfortunately, as shown in **[9,10]**, inverse kinematics has the non convexity property and therefore does not permit direct learning of the inverse mapping. Nevertheless, as noted by Bullock et al **[5]**, it is possible to transform the non convex problem of inverse kinematics learning into a convex problem by spatially localizing the learning task: within the vicinity of each robot configuration reached during learning, inverse kinematics is actually convex. Thus, inverse kinematics of a redundant system can theoretically be accomplished properly by learning an inverse mapping if a spatially localized learning algorithm is employed. Following previous works by Bullock et al., **[5]** and Guenther & Micci Barreca, **[7]** in the direct inverse model described later in this paper, learning generalizes to all spatial directions at each sampled joint configuration; this is because the model learns a directional mapping that is an approximation to the Jacobian pseudoinverse at each joint configuration, and the approximate Jacobian pseudoinverse learned for one movement direction can be used for all other movement directions. It's also important to note that the directional mapping learned by the neural network model presented in this paper is locally linear, even for redundant systems. This means that if one only considers a small region of joint space, the set of joint velocity vectors that produce a desired spatial velocity is convex. The radial basis network described in subsequent sections utilizes different parameters in different regions of the workspace (corresponding to different radial basis functions) and smoothly interpolates between these parameter sets. It is also important to note that systems that use directional mappings as the system presented in this paper, can successfully reach targets even if the directional mapping contains a large amount of error. Therefore, any residual error that might exist; e.g. from assuming linearity over too large a region of the workspace, will not prevent the system from reaching targets, but will instead only lead to curvature in the movement trajectories.

2 Learning Regularized Inverse Kinematics Solutions. HypBF Networks

2.1 Regularization Networks. Hyper Basis Functions Networks.

The main hypothesis underlying this paper is that inverse kinematic learning can be accomplished by using a special kind of multivariate function approximation realized as HyperBF networks **[15]**. It has been commented previously that learning a smooth inverse mapping such as required by inverse kinematics is clearly an ill-posed task, in the sense that the information in the data is not sufficient to reconstruct uniquely the mapping at points where data are not available. A priori assumptions about the mapping are needed to make the problem well-posed. One of the simplest assumptions is

that the mapping is smooth: small changes in the inputs cause a small change in the output. Techniques exploiting smoothness constraints in order to transform an ill posed problem into a well posed one are well known under the term of regularization theory. In **[15]** it was showed that the solution to the approximation problem given by regularization theory can be expressed in a class of multilayer networks that they called regularization networks or Hyper Basis Functions networks. The main result of these authors is that the regularized solution is equivalent to an expansion of the solution in terms of a certain class of functions:

$$f(x) = \sum_{i=1}^{N} c_i G(x, \xi_i) \tag{3}$$

where *G(x)* is one such function and the coefficients c_i satisfy a linear system of equations that depend on the examples of data to be approximated. Under relatively broad assumptions, the Green's function *G(x)* is radial ant therefore the approximating function becomes

$$f(x) = \sum_{i=1}^{N} c_i G\left(\|x - \xi_i\|^2\right) \tag{4}$$

which is a sum of radial functions, each with its center ξ_i on a distinct data point. The number of radial functions, and corresponding centers, is the same as the number of examples (*N*). Nevertheless the network associated with equation [4] can be made more general in terms of the following extension

$$f^*(x) = \sum_{\alpha=1}^{n} c_\alpha G\left(\|x - t_\alpha\|_W^2\right) \tag{5}$$

where the parameters t_α, that we will call centers, and the coefficients c_α are unknown, and are in general much fewer than data points. The norm is a weighted norm

$$\|x - t_\alpha\|_W^2 = (x - t_\alpha)^T W^T W (x - t_\alpha) \tag{6}$$

where *W* is an unknown square matrix and superscript T indicates transpose. In the simple case of a diagonal *W* the diagonal elements assign a specific weight to each input coordinate, determining in fact the units of measure and the importance of each feature. Notice that a set of Gaussian *G* functions with a diagonal *W* are equivalent to the same gaussians with their own spreads σ (i.e. calling $w_{ll} = 1/\sigma_l$ and doing $w_{lk} = 0$ if $l \neq k$)

In this framework, the stage of learning is simply the stage of estimating from the data the values of the parameters in the representation that has been showed above. Iterative methods can be used to find the optimal values of the various sets of parameters , the c_α, the *W* matrix and the t_α, that minimize an error functional on the set of

examples. Steeprest descent is the standard approach that requires calculations of derivatives. The error functional is defined as

$$H\left[f^*\right] = H_{c,t,W} = \sum_{i=1}^{N} \left(\Delta_i\right)^2 \tag{7}$$

$$\Delta_i \equiv y_i - f^*\left(x\right) = y_i - \sum_{\alpha=1}^{n} c_\alpha G\left(\left\|x - t_\alpha\right\|_W^2\right) \tag{8}$$

and the parameters that minimize the error functional are regarded as the coordinates of the stable fixed point of the dynamical system described by equations (9) – (11):

$$\dot{c}_\alpha = -\omega \frac{dH\left[f^*\right]}{dc_\alpha} \tag{9}$$

$$\dot{t}_\alpha = -\omega \frac{dH\left[f^*\right]}{dt_\alpha} \tag{10}$$

$$\dot{W} = -\omega \frac{dH\left[f^*\right]}{dW} \tag{11}$$

where ω is the learning parameter.

The interpretation of the network described above is the following. After learning, the centers of the basis functions are similar to prototypes, since they are points in the multidimensional input space. Each unit computes a weighted distance of the inputs from its center, that is a measure of their similarity, and applies it the radial function. In the case of the Gaussian a unit will have maximum activity when the new input exactly matches its center. The output of the network is the linear superposition of the activities of all the basis functions in the network. During learning the weights *c* are found by minimizing a measure of the error between the network's prediction and each of the examples. At the same time the centers of the radial functions and the weights in the norm are also updated during learning. Moving the centers is equivalent to modifying the corresponding prototypes and corresponds to task dependent clustering. Finding the optimal weights *W* for the norm is equivalent to transforming appropriately, for instance scaling, the input coordinates and corresponds to task dependent dimensionality reduction.

2.2 HypBF implementation for learning Inverse Kinematics of a robot manipulator

The neural network described in this section (Figure 3) was trained to learn the directional mapping described in past paragraphs and then was tested on the robotic platform described below to demonstrate some of the desirable properties of the adaptive control scheme proposed in this paper. The mapping from the desired movement direction vector Δx to the joint rotation vector $\Delta\theta$ is formed in the model is formed according to the equation (12):

$$\Delta\theta = A(\theta)\Delta x \tag{12}$$

The elements of the matrix $A(\theta)$ are the outputs of a regularization HyperBF network and are calculated according to the following equations:

$$a_{ij} = \sum_k \frac{g_k^a}{\sum_l g_l^a(\theta)} w_{kji}^a \tag{13}$$

where w^a_{kij} are scalar parameters and the basis functions are Gaussian:

$$g_k^a(\theta) = \exp\left(-\frac{1}{2}\sum_l \left(c_{kl}^a(\theta)\right)^2 \right) \tag{14}$$

$$C_{kl}^a = \frac{\theta_l - \mu_{kl}^a}{\sigma_{kl}^a} \tag{15}$$

where μ^a_{kl} and σ^a_{kl} are parameters corresponding to the mean and variance of basis function k along dimension l. Parameters μ^a_{kl} correspond to centers t_α of subsection 2.1 and variances σ^a_{kl} corresponds to diagonal elements of matrix W, where $w_{ii} = 1/\sigma^a_{ki}$. The parameters w^a_{kij}, μ^a_{kl}, σ^a_{kl}, are learned using gradient descent during the action – perception cycle used to train the model. The action – perception cycle is induced by instanting random joint velocity vectors $\Delta\theta^B$ (where the subscript B denotes a babbled movement, Figure 1) . For the parameters used to produce a_{ij}, the cost function or error functional that is minimized by gradient descent is:

$$H = \sum_i \left(\Delta\theta_i^B - \Delta\theta_i\right)^2 \tag{16}$$

, where the $\Delta\theta_i$ form the joint rotation vector calculated from the HyperBF network outputs according to equation (12). The values a_{ij} form the approximate Jacobian pseudoinverse learned by the network.

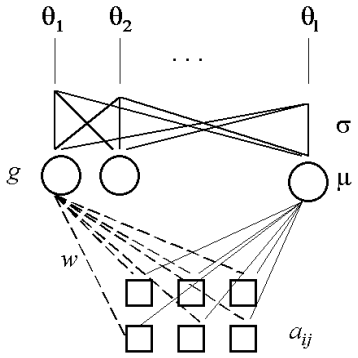


Figure 3. A HyperBF network for computing the elements of approximate Jacobian pseudoinverse.

In Figure 3, the data are a set of joints increments $\Delta\theta^B$ generated randomly during training, equivalent to babbled movements. These joints rotations are carried out from certain joint configuration denoted by θ, that is the input to the HyperBF network. Babbled joint rotations induce spatial displacements of the end effector denoted by Δx, displacements that are measured by our stereohead vision system. HyperBF network computes an approximate Jacobian pseudoinverse that is used to compute an estimation of $\Delta\theta^B$ $(\Delta\theta)$. With this estimation we construct the functional of error or cost function H to derive the changes in parameters σ,μ and w of the network. In other words, the network learns to compute a linear approximation of Jacobian pseudoinverse at each joint configuration θ.

$$\varepsilon_i = \left(\Delta\theta_i^B - \Delta\theta_i\right) \tag{17}$$

$$h_k = \frac{g_k^a(\theta)}{\sum_l g_l^a(\theta)} \tag{18}$$

$$\frac{dw_{kij}^a}{dt} = -2\omega\varepsilon_i \Delta x_j h_k \tag{19}$$

$$\frac{d\mu_{kl}}{dt} = -2\omega h_k\left(1-h_k\right)\frac{\|c_k\|}{\sigma_{kl}}\sum_{i,j}\varepsilon_i w_{kij}^a \Delta x_j \tag{20}$$

$$\frac{d\sigma_{kl}}{dt} = -2\omega h_k\left(1-h_k\right)\frac{c_{kl}\|c_k\|}{\sigma_{kl}}\sum_{i,j}\varepsilon_i w_{kij}^a \Delta x_j \tag{21}$$

3 Visuo-Motor System for tracking and location

The visuo-motor system implemented for testing the described neural architecture for reaching model follows neuro-biological models proposed in the CNS (*Cognitive and Neural Systems*) research group at the Boston University. Grossberg, Bullock and others, published some models of the animal neural system related with reaching process. Adaptation of these models to redundant robotic platforms has permitted to develop a neural control architecture for tracking and location objects. A colour-based image visual processing algorithm together with an anthropomorphic robotic stereohead project the extracted features from the two images over the head motor joints. The relationship between both representation spaces is carried out by means of VAM (Vector Associative Maps) adaptive algorithms **[13]**. They consist of self-organizing neural models that quickly project sensorial onto motor information in robust way. All the necessary knowledge of the robotic platform is learned by means of action – reaction cycles from visual-motor trials. This neural architecture has been developed integrating a set of neural networks, of some discovered biological function, carries out by the animal neural system. This visual-motor architecture contains two main modules corresponding with the interconnected processes of spatial internal representation module and the stereohead controller.

3.1. Spatial Internal Representation Module

This module carries out an internal Cartesian representation on a body-centred frame of the selected objective (robotic arm end-effector or object) position. This algorithmic module has been developed, starting from neural network models of how the human brain learns spatial representation, controlling sensory-guided and memory-guided eye and limb movements. This spatial representation is expressed in both head-centred and body-centred coordinates, because the eyes move within the head, whereas the head, arms, and legs move with respect to the body. In a binocular system, it is possible to represent the position of an objective from the geometrical properties of the head: version (ϕ), vergence (θ) and elevation (γ), as figure 4 shows. Geometrical relationship between P(x,y,z) and P(ϕ, θ, γ) could be found in **[4]**.

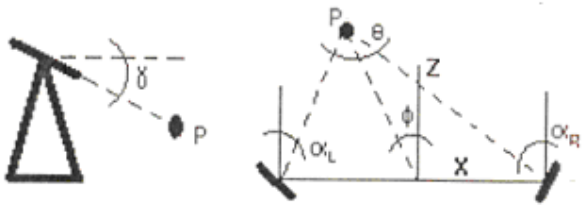


Figure 4. Stereohead reference system for object representation (P)

3.2. Stereohead controller

This module implements a visuo-motor control for the stereohead ocular joints, moving the neck and ocular joints of the stereohead. It places the stereohead joints in a situation of symmetric vergence, which is the most favourable position for visual processing and position representation. For the control of the ocular joints an AVITE algorithm **[6]** has been implemented. Figure 5 describes the main components of the neural head controller.

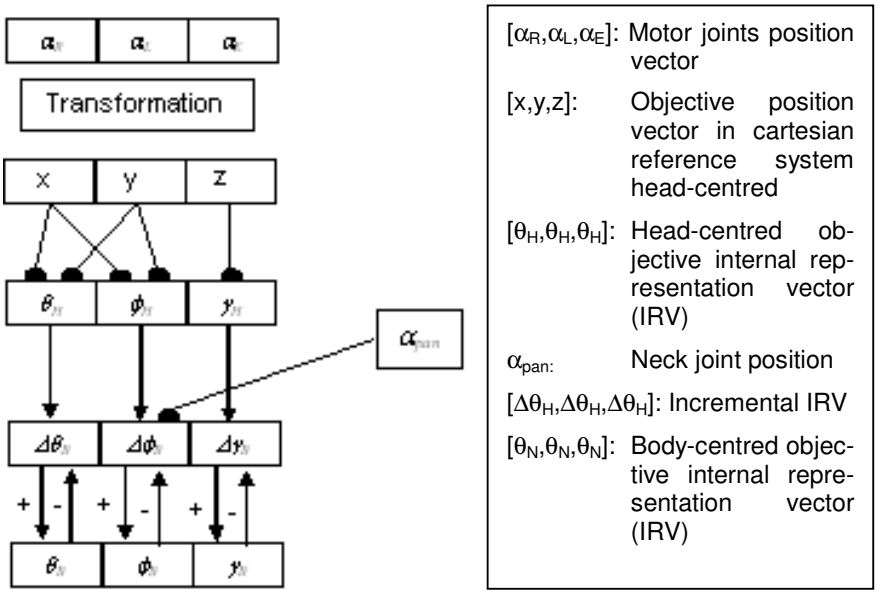


Figure 5. Visuo-Motor controller for objective representation

The neck controller has the function of maintaining the head structure in the best position in order to perform the visualization of the targets. The optimal position is that in which the head has $\phi_H=0$. To solve this problem, a self-organizing neural network based on AVITE model has been used. The mapping between version variable (ϕ_H) and a neck compensation variable (α_{pan}) is established linearly by means of an adaptive weight. In the learning phase, panoramic movements with random values of incremental rotation angles are generated. Then, the ocular controller module fix the target and the spatial representation module calculates the new incremental values for ($\Delta\phi_H$) with respect to reference situation ($\phi_H=0$).

4 Experimental Robotic Platform

For testing the behaviour of the described neural model, a robotic platform composed by the LINCE anthropomorphic robotic stereohead, one industrial robot arm from ABB Company and two colour cameras for object detecting has been employed. A client-server architecture has been designed for establishing the TCP/IP-based communications between the robot arm controller, the robot head controller and the active vision system. In figure 6 a picture of the robotic installation is shown.

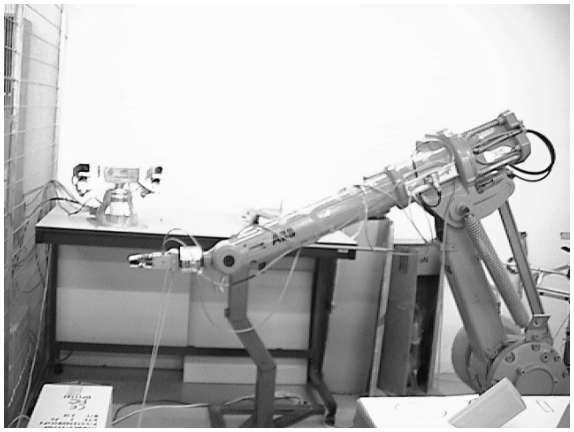


Figure 6. NEUROCOR robotic platform

In order to project the spatial coordinates for the object and the end-effector over the robot arm joint positions, two coordinate frames have been considered, one refereed to the body of the stereohead and other which is incorporated to the robot arm controller. The implemented neural controller receives information from both systems and generates the neural weights map to project the difference vector ($\Delta\theta$) from the joint positions over the difference vector (ΔP) from the spatial positions. Thus, in figure 7, the scheme of the arm-head relative coordinate frame is represented.

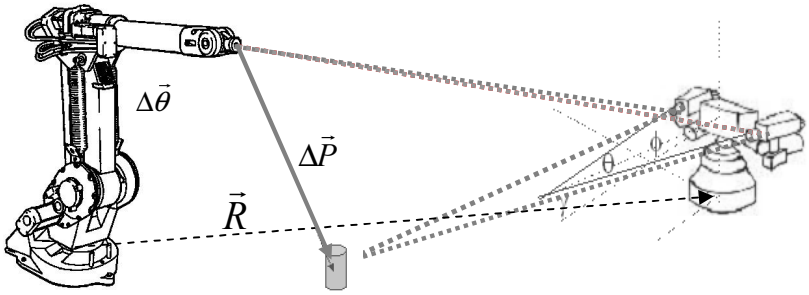


Figure 7. Scheme of the relative reference system of the robotic platform

5 Results

The implementation of the proposed system has been carried out in the below robotic platform. The obtained results have allowed the verification and tests for the capability of the proposed neural architecture by adapting the neural map configuration to the dynamic environment with redundant robotic systems. The developed experiments have been focused to analyze the accuracy of the model for reaching tasks as a function of error thresholds, its robustness when perturbations are considered, and the capability for tracking moving objects.

The expression for the signal GO(t) is given by:

$$GO(t) = \frac{KGo \cdot t^3}{300 + t^3} \tag{22}$$

while the expression for final error E(t) in each instant is:

$$E(t) = \sqrt{\sum_{i=1}^{3} \left(T\arg et_i(t) - CurrentPosition_i(t) \right)} \tag{23}$$

where, 'i' indicates the x, y or z coordinate into the absolute reference system. The configuration data for the developed experiments are shown in Table 1. In all cases the arm-head relative position, defined by the R vector has been [-3000,0,800] the KGo gain = 20, the number of trials for the neural network is 3000, the dimension for the weights matrix is [3x3x250] and for the variance and center position of the gaussians functions, [3x250] values have been considered for each matrix. The position of the robot arm end-effector is obtained for the image processing algorithm by means o a green label placed over the wrist, while the object (a small sphere) is detected by yellow colour. For the robot arm, only the base, shoulder and elbow joints of the industrial robot arm have been taking into account, due the rest of the joints intervene in the final orientation of the tool, which is far of the objectives for this work.

Table 1.Configuration data for the experiments

	Umbral error	Initial joints	Initial XYZ	Target XYZ	Perturbation
P1	1	10,-50,15	2623,-66,742	2150,1250,150	NO
P2	5	10,-50,15	2623,-66,742	2150,1250,150	NO
P3	10	10,-50,15	2623,-66,742	2150,1250,150	NO
P4	1	10,5,-5	1855,-202,503	Three points	NO
P5	5	10,-40,10	2736,-46,944	2150,1250,400	1600,-450,-200
P6	5	10,-20,15	2129,-153,542	Slow motion	YES
P7	1	10,-20,15	2129,-153,542	YZ circular trajectory	YES

For the verification of the reaching capabilities of the model the P1, P2 and P3 have been carried out, in absence of perturbations. The obtained results are represented in Figures 8 to10.

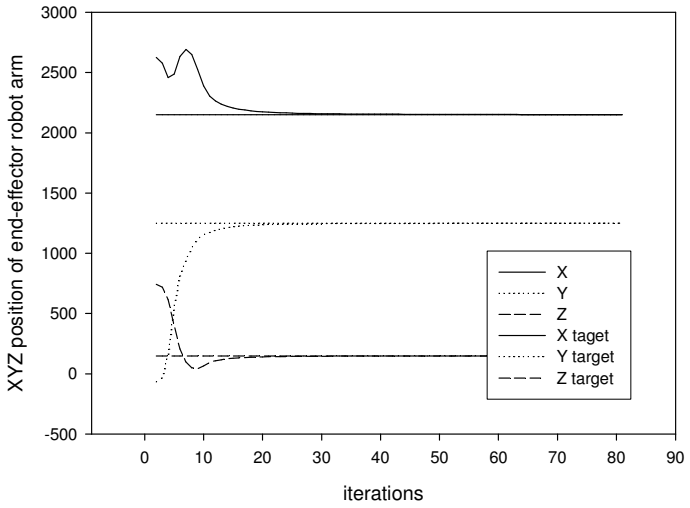


Figure 8. Evolution of end-effector in the experiment P1

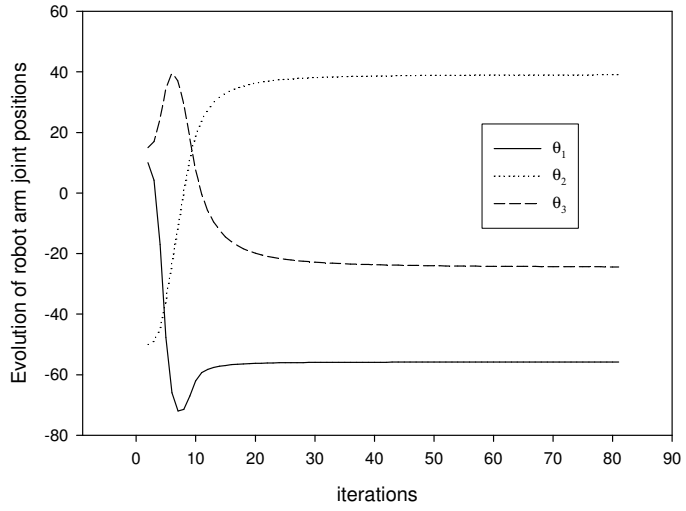


Figure 9. Evolution of robot arm joints in the experiment P1

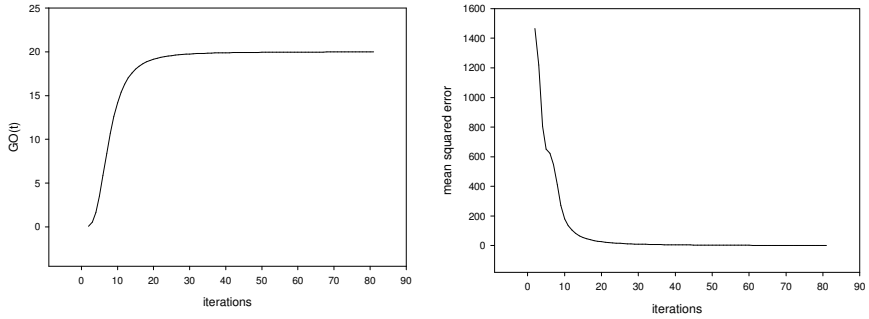


Figure 10. Evolution of GO(t) and final error in the experiment P1

For analyzing the relationship between the iterations number to reach the object and the required precision, P2 and P3 experiments have been carried out. The main results are compared and shown in Table 2.

Table 2. Results for velocity vs. accuracy in reaching tasks

Nº Experiment	Final Error	Iterations	Final joints position
P1	0,9916	81	[-55.80, 39.02, -24.37]
P2	4,8596	40	[-55.86, 38.61, -23.70]
P3	9,5301	30	[-55.95, 38.10, -22.85]

The representation of the curved trajectory which is a particular characteristic of the HRBF neural model has been obtained in the P4 experiment for three different spatial positions of the sphere. Their deviations with respect to the optimum linear trajectory are represented in figure 11.

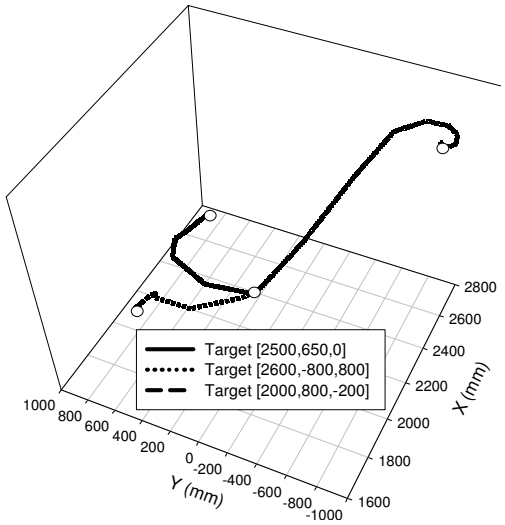


Figure 11. Curved trajectory for reaching with HRBF model in the experiment P4

In order to evaluate the behaviour of the HRBF neural model with the robotic platform a reaching task with random perturbations has been experimented and analyzed. For the experiment P5, the system has been leaded to reach the sphere but in some instant, the object is quickly changed and the variation of the end-effector trajectory is represented in figures 12 to 14.

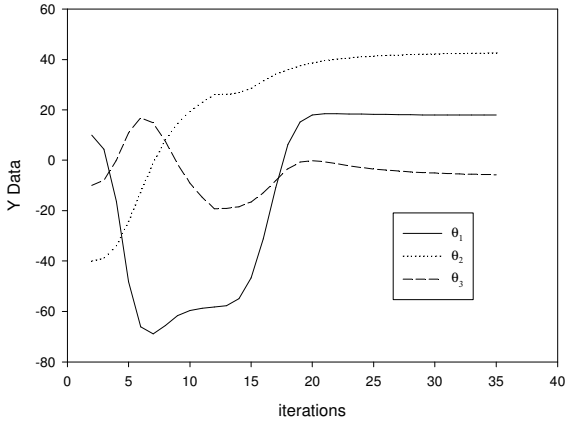


Figure12. Joint positions evolution in presence of perturbations in the experiment P5

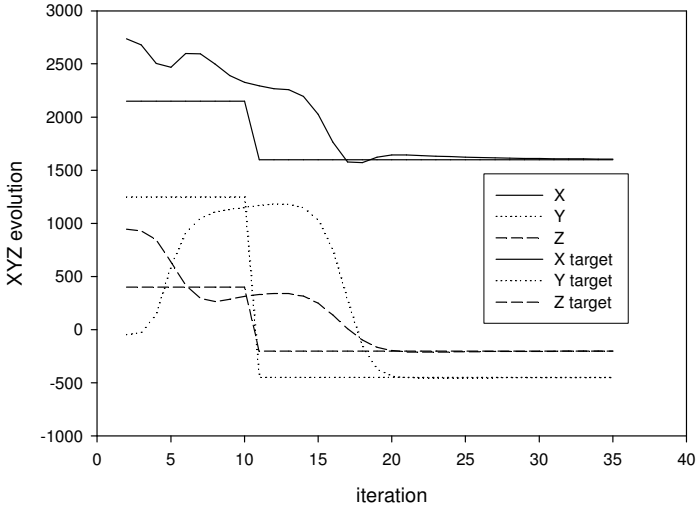


Figure 13. Target and end-effector evolution in the experiment P5.

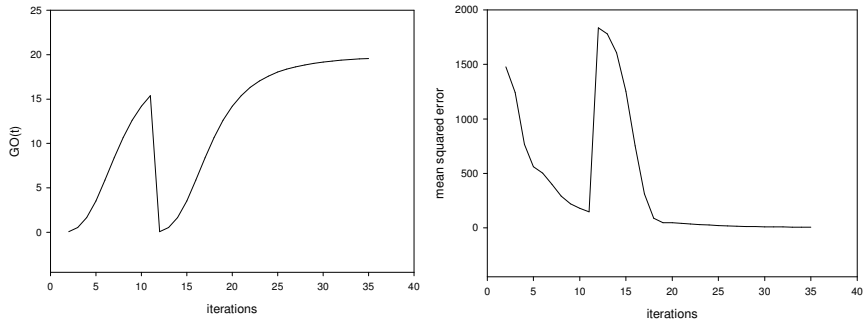


Figure 14. Evolution of GO(t) and final error in the experiment P5

Finally, the neural model HRBF has been tested for tasks involved tracking objects. For this case, the behaviour has been analyzed for both random (P6 experiment) and circular (P7 experiment) movements of the object. For this one, a circular trajectory for the object in the YZ plane has been generated helped by a mechanical device like a wheel. The obtained results are represented in figures 15 to 20.

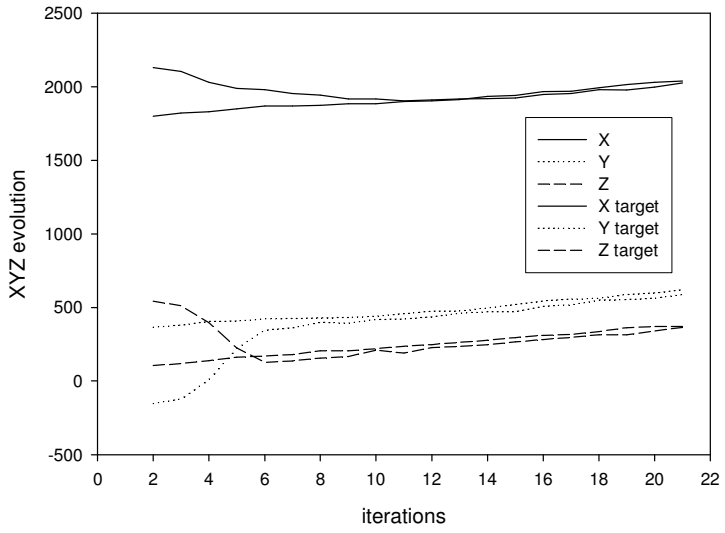


Figure 15. Target and end-effector evolution in the experiment P6.

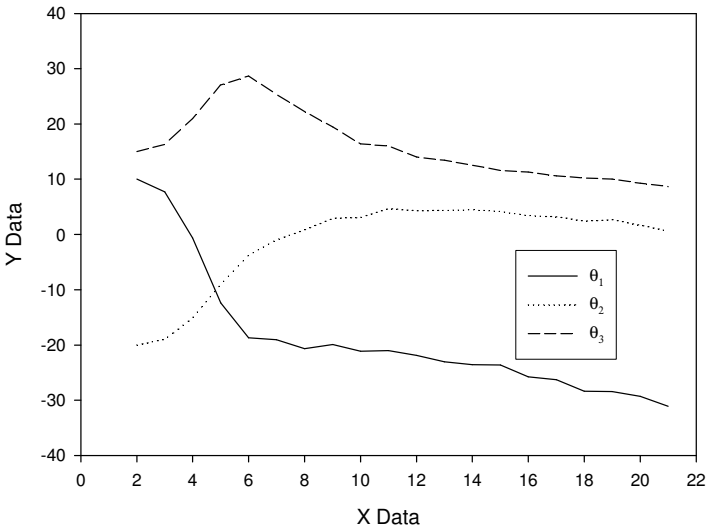
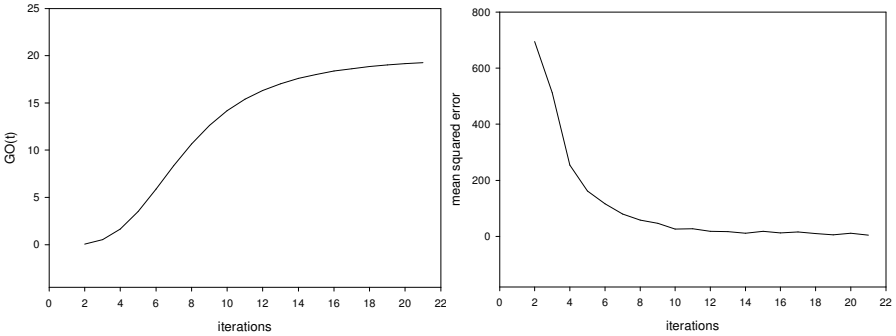


Figure 16. Joint positions evolution in tracking task in the experiment P6



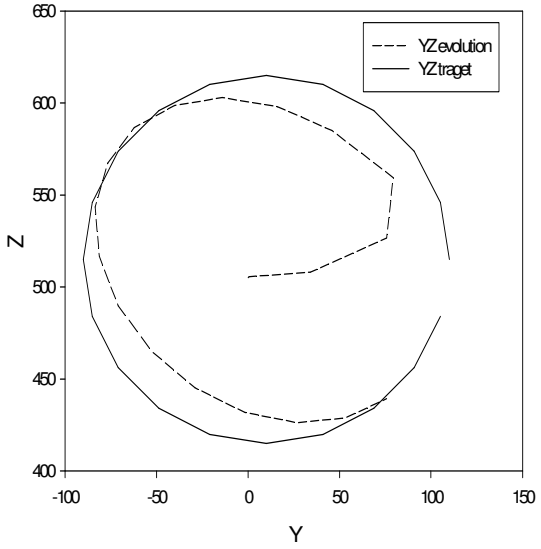


Figure 18. Target and end-effector evolution in the experiment P6.

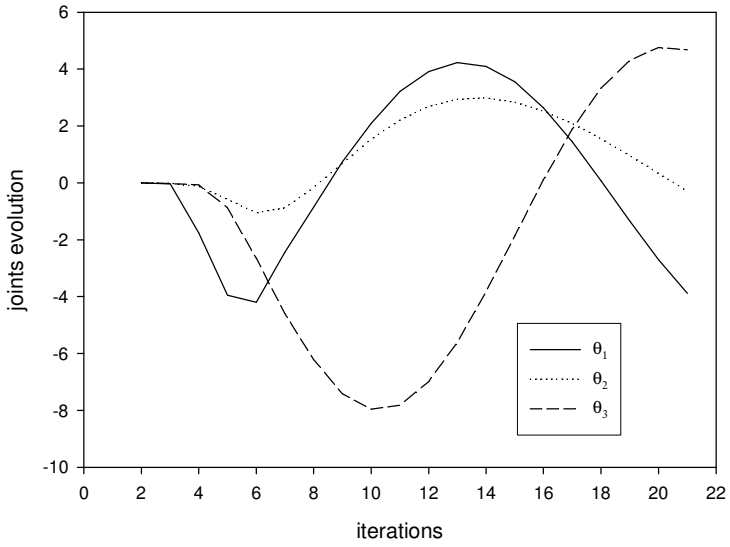


Figure 19. Joint positions evolution in tracking tasks in the experiment P7

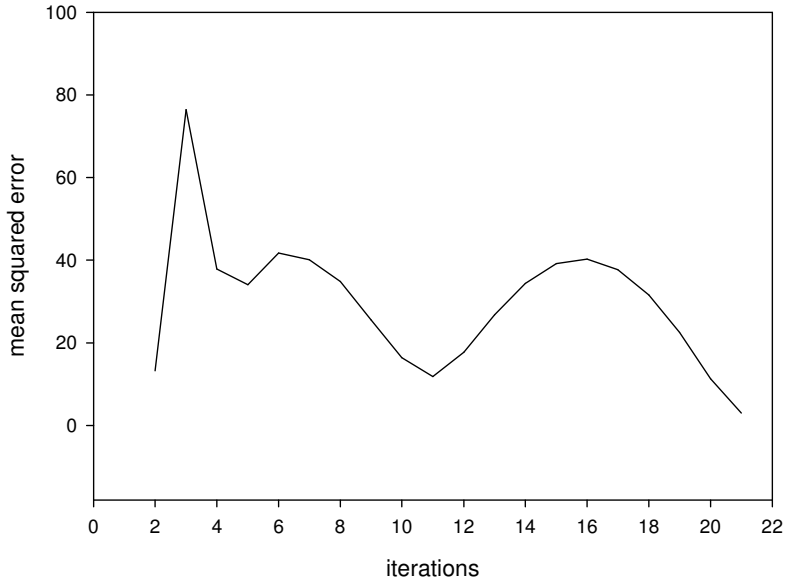


Figure 20. Final error evolution in the experiment P7

6 Conclusions

In this paper a robust visuo-motor architecture applied to redundant robots for reaching tasks has been implemented and their results analyzed. The proposed neuro-controller is based in AVITE neural models for the visuo-motor control of anthropomorphic stereoheads foveating objects and in HypBF neural networks for solving the inverse kinematic of redundant robot arms. A head-arm robotic platform over a client-server architecture for TCP/IP communications has allowed to test the characteristics of the proposed architecture focused to different operations: reaching with and without perturbations in the object position, foveating objects, and tracking 3D trajectories. The combination of the robustness and accuracy of the HypBF model and the fast computing for the AVITE model, together the integration of both neural networks for the learning and performance phases gives a solution for reaching applications when the precision is a required parameter. Several configurations and sceneries have been carried out for reaching a small sphere by means of colour-based visual algorithm and, in all the experiments, the minimum error has been found for a reduced number of movements for the robot arm.

References

[1] E.W. Aboaf, C.G. Atkeson, D.J. Reinkensmeyer. (1988). Task level robot learning. In Proceedings of the IEEE international conference on Robotics and Automation, Philadelphia, PA. pp 1309-1310.

[2] E.W. Aboaf, S.M. Drucker, C.G. Atkeson,. (1989). Task level robot learning: Juggling a tennis ball more accurately. In Proceedings of the IEEE international conference on Robotics and Automation, Scottsdale, AZ. pp 1290-1295.

[3] J. Ballieul, J.M. Hollerbach. R.W Brocket. (1984). Programming and control of kinematically redundant manipulators. In Proceedings of the 23rd IEEE Conference on Decision and Control, pp 768 – 774.

[4] D. Bullock, S. Grossberg.(1989). VITE and FLETE: Neural modules for trajectory formation and tension control. In W. Hershberger, (ed.): Volitional Action, Amsterdam: North-Holland, pp.253-297.

[5] D. Bullock, S. Grossberg, F.H. Guenther (1993). A self organizing neural model of motor equivalent reaching and tool use by a multijoint arm. Journal of Cognitive Neuroscience, 5(4),pp 408 – 435.

[6] S. Grossberg, F. Guenther, D. Bullock, D. Greve (1993), "Neural representation for sensory-motor control II. "Learning a head-centered visuomotor representation of 3D target positions". Neural Networks 6, pp. 43-67.

[7] F.H. Guenther, D. Micci – Barreca. (1997). Neural models for flexible control of redundant systems. In P. G. Morasso & V. Sanguinetti (Eds): Self – Organization, Computational Maps, and Motor Control. Elsevier, North Holland Psychology series, pp 383 – 421.

[8] J.M. Hollerbach, K.C. Suh. (1985). Redundancy resolution of manipulators through torque optimization. In Proceedings of the IEEE International Conference on Robotics and automation, pp 1016 – 1021.

[9] M.I. Jordan. (1990). Motor learning and the degrees of freedom problem. In M. Jeannerod (Ed), Attention and performance XIII: Motor representation and control, Hillsdale NJ: Erlbaum, pp 796 - 836

[10] M.I. Jordan, D.E. Rumelhart. (1992). Forward models: Supervised learning with a distal teacher. Cognitive Science, 16, pp 307 – 354.

[11] C.A. Klein, C. Huang. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. IEEE Transactions on Systems, Man and Cybernetics, SMC – 13(2), pp 245 – 250.

[12] A. Liegeois. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Transactions on Systems, Man and Cybernetics, SMC – 7(12), pp 868 - 871

[13] J. López-Coronado, J.L. Pedreño-Molina, A. Guerrero-González, P. Gorce.(2002) A neural model for visual-tactile-motor integration in robotic reaching and grasping tasks. Robotica, 20, pp 23-31

[14] F.A. Mussa – Ivaldi, N. Hogan. (1991). Integrable solutions of kinematic redundancy via impedance control. International Journal of Robotics Research, 10, pp 481 – 491.

[15] T. Poggio, F. Girosi. (1989). A theory of networks for approximation and learning. AI Memo No. 1140, Massachustts Institute of Technology.

[16] E. Saltzman, S.J.A. Kelso. (1987). Skilled Actions: A task dynamic approach. Psychological Review, 94(1), pp 84 – 106.

[17] D.E. Whitney (1969). Resolved motion rate control of manipulators and human prostheses. IEEE Transactions on Man machine Systems, 10(2), pp 47 – 53.