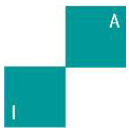
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial
ISSN: 1137-3601
revista@aepia.org
Asociación Española para la Inteligencia Artificial
España

Tello Gamarra, Daniel Fernando; de Souza Leite Cuadros, Marco Antonio
Forward Models for Following a Moving Target with the Puma 560 Robot Manipulator
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 18, núm. 56, 2015, pp. 31-42
Asociación Española para la Inteligencia Artificial
Valencia, España

Inteligencia Artificial 18(56) , 31-42
doi: 10.4114/ia.v18i56.1105



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Forward Models for Following a Moving Target with the Puma 560 Robot Manipulator

Daniel Fernando Tello Gamarra[1] and Marco Antonio de Souza Leite Cuadros[2]

[1]Universidade Federal de Santa Maria (UFSM)
Control and Automation Engineering, Santa Maria, Rio Grande do Sul, Brazil.
daniel.gamarra@ufsm.br

[2]Instituto Federal do Espirito Santo (IFES)
Serra, Espirito Santo, Brazil.
marcoantonio@ifes.edu.br

Abstract This paper describes how a forward model could be applied in a manipulator robot to accomplish the task of following a moving target. The forward model has been implemented in the puma 560 robot manipulator in simulation after a babbling motor phase using ANFIS neural networks. The forward model delivers a rough estimation of the position in the operational space of a moving target. Using this information a Cartesian controller tracks the moving target. An implementation of the proposed architecture and the Piepmeir algorithm for the problem of following a moving target is also shown in the paper. The control architecture proposed in this paper was also tested with MLP and RBF neural networks. Results and simulations are shown to demonstrate the applicability of our proposed architecture for tracking a moving target.

Keywords: Forward Models; Visual Servoing; Cartesian Control; Motor Babbling; Neural Networks.

1 Introduction

Studies in the field of neuroscience would suggest that human beings create internal models for accomplishing different tasks that involve their motor apparatus and sensory systems in [23] and [11]. Internal models are divided in forward models and inverse models. Forward models can predict sensory consequences from efference copies of issued motor commands as stated in [11].

The process of creation of forward models starts in the first periods of human development, and emerges through self-exploration and interaction with the external world. The stage of self-exploration of our own kinematics and sensory system is executed in a process called "motor babbling", furthermore, some works in the field of developmental psychology in [21] and [19] pointed out the existence of a rudimentary form of eye-hand coordination that is developed during the exploration of the newborn space and kinematics in [21] . Roboticists in particular have started to seek inspiration from biological systems and how they deal with complexity of the world in which they live as stated in [14], as a consequence, some works involving the use of forward models have been published. For instance, Nori in [15] describes a motor module system based in forward and inverse models that map from motor commands to the corresponding movements and vice versa, relating the arm positions and the torques, after a learning phase of the forward and inverse models, the robot generalizes this information when it has to pick up new objects.

In ([12]) and ([6]) a robot manipulator with a stereo vision system, after a babbling motor phase builds a forward model that embedded information from its joint configurations and its end effector position in the image field of view. The forward model was employed by the robot for estimating an image Jacobian used in a visual servo controller. Also Gamarra developed an application of forward models for a ballistic reaching task in

ISSN: 1988-3064(on-line)

([5]). Saegusa as well in [16], proposed a developmental approach of body definition without prior knowledge on kinematics, dynamics, and body appearance. The visuomotor correlation allows the robot to define its own body through sensorimotor exploration. Furthermore, Walter in [22] proposed an approach based in the construction of visuomotor maps, a kind of forward models, Walter mapped a target position in the image plane using self-organizing maps, our approach has basic differences with respect to Walter's, we have used three different kind of neural networks (ANFIS, MLP and RBF); the target is moving and the mapping is made from the image plane to the Cartesian space.

A new application of forward models for the task of following a moving target is proposed in this paper, traversing a developmental robotics roadmap, the forward model was constructed using the stored information from a babbling motor phase, in which, the joint angles, end effector coordinates and image features data of the robot were saved, afterwards, is initialized a task that consists in following a moving target; in a first phase, the information of the forward model is embedded in ANFIS neural networks, the forward model employing the target coordinates in the image field of view delivers the position in Cartesian space of the target. In a second phase, the desired position will be the input to a Cartesian controller utilized for following a moving target.

The remainder of the paper is as follows. In the second section the experimental platform setup is described; the third section shows the proposed architecture using forward models designed for following a moving target; in the fourth section are described the algorithms used in this work; in the fifth section the motor babbling process is explained; in the sixth section the forward model creation is detailed; the seventh section presents the results of the controllers performances in following a moving target task, finally, conclusions and planned future works are discussed in the last section.

2 Experimental Platform Setup

This section describes briefly the experimental platform setup for the proposed system, figure 1 depicts the experimental simulation setup for the proposed system. The simulations developed in this paper were done using the puma 560 robot manipulator. The unimation puma 560 is an industrial manipulator based in revolute joints with six degrees of freedom. Three big motors provide the waist, shoulder and elbow control. As well as three small motors provide the position and orientation of the robot wrist as refered in [1]. The simulations shown in this article used the matlab robotics toolbox created by Corke and described in [3]. The epipolar geometry toolbox build by Mariottini in [13] was used to create a scenario with multiple cameras and manipulate their visual information.

Figure 1 shows the simulated robot and the cameras in three dimensional ($3D$) space and also the camera captured images of the end-effector and target in two dimensional ($2D$) space. A Neural network will be employed in order to construct a forward model. The Matlab implementation of ANFIS neural networks from the fuzzy logic toolbox was also used in the simulations, three different neural networks will be tested separatly for the forward model construction: ANFIS, MLP and RBF neural networks. The neural networks Matlab toolbox for the MLP and RBF networks was employed. In all the simulations run in the paper, a Gaussian noise was introduced in the vision system with a covariance of 0.5. Finally, visual servoing and a Cartesian controller using the forward model will be employed in order to achieve the tracking of the target.

3 Proposed Architecture Description Using Forward Models

Figure 2 depicts the new control architecture proposed on this paper. This architecture could be resumed as a Cartesian Controller for tracking a moving target that uses a forward model previously constructed by a neural network. A forward model was constructed after a motor babbling phase and encodes the relations between the end effector Cartesian position and the end effector coordinates in the stereo vision system. ANFIS, MLP and RBF neural networks are used for the forward model creation. Initially the forward model was constructed using an ANFIS neural network, but in order to generalize the method the ANFIS was substituted for the MLP neural network and posteriorly for the RBF neural network.

Once constructed the forward model, the control system architecture works in the scenario in which a moving target is detected using the vision system. The vision module capture images of the moving target as inputs and delivers as output the image features of the moving target. The coordinates of the target in both cameras are send to the forward model as inputs, the forward model delivers as output the Cartesian position of the moving target, the information of the desired Cartesian position is used by the Cartesian controller as input or set point for following a moving target and the outputs of the Cartesian controller are the joint angles for the manipulator. In order to compare the control architecture that is proposed in this paper the Piepmeir's algorithm for tracking a moving target was also implemented.

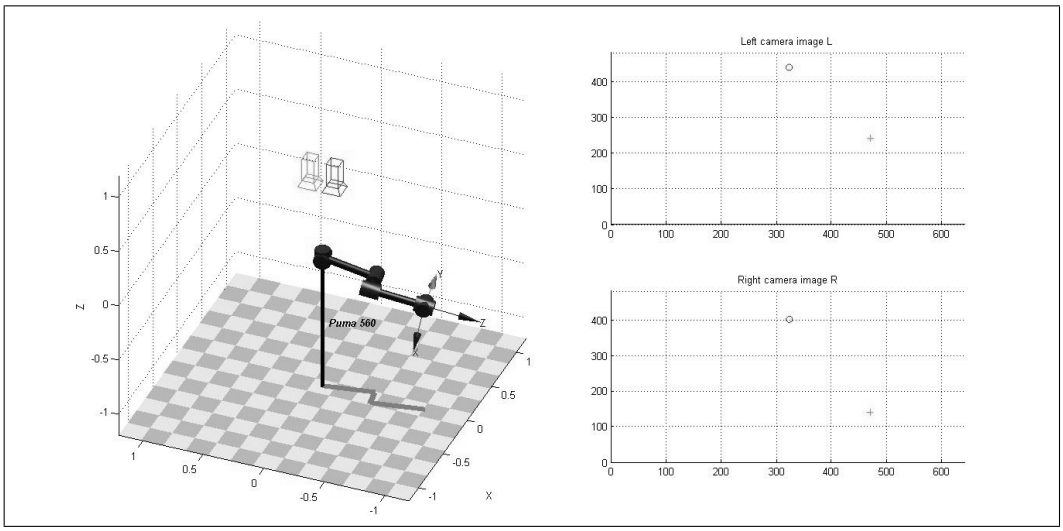


Figure 1: Simulation System Setup. The left subplot shows the position of the robot and stereo cameras at the beginning of the tracking in $3D$ space. The right top and right bottom subplots show the target (cross point) and end-effector position (circle point) in the captured image $2D$ space.

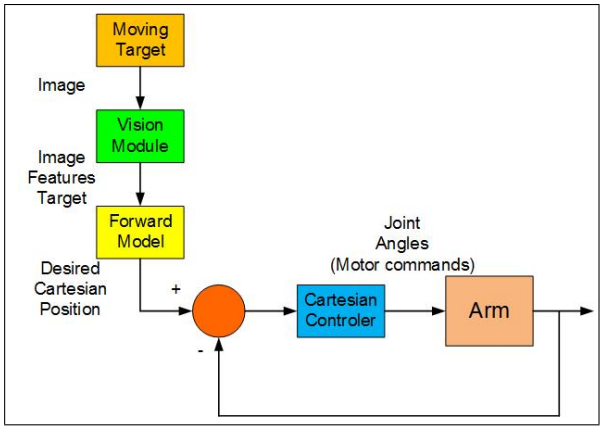


Figure 2: Proposed architecture.

4 Theoretical Background

4.1 Cartesian Space Control

In Cartesian space control the sensed position of the manipulator is immediately transformed by means of the kinematic equations into a Cartesian description of position. This Cartesian description is then compared to the desired Cartesian position in order to form errors in Cartesian space as stated in [4].

Based on the equations shown in Sanchez-Sanchez's paper in [17], it will be described briefly the Cartesian controller. The robot end-effector moves in $3D$ space, so a function that relates the end-effector position and orientation X of the end effector and the robot joints variable q is given by:

$$X = f(q) \tag{1}$$

If we derive partially this equation the inverse Jacobian matrix ($J^{-1}(q)$) is obtained. If the Jacobian matrix is not square, we could use the pseudo-inverse matrix, the joint velocities for the manipulator are calculated given the equation:

$$\dot{q} = \lambda \times J^{-1}(q)(\dot{X} - \dot{X}_d) \tag{2}$$

Where $\dot{X}_d$ represents the Cartesian desired position and λ is the proportional gain controller.

4.2 Visual Servoing for Following a Moving Target

Visual servoing is defined as control based on feedback of visual measurements as referred in [7]. There are two categories for implementing vision-based control techniques, those that realize visual servoing in operational space, termed as position-based visual servoing (PBVS) and those that realize visual servoing in the image space, known as image-based visual servoing ([18])(IBVS). The controller implemented in this paper would be classified according to Siciliano in [18]) and Hutchinson in [7] as an Image Based Visual Servoing technique, so it just will use the images captured for the camera for the Visual Servoing, more specifically the stero-vision system for our application. The algorithm used for the visual servoing in this work is the one proposed by Piepmeier in [10] and [9]. Piepmeier used a dynamic quasi-Newton method that minimizes a time-varying objective function based on errors in the image plane as referred in [9]. The error function, for a moving target is defined as the difference of the position in the image plane of the moving target and the end effector

$$f(\theta) = y(\theta) - y^* \tag{3}$$

Where θ represents the joint angles, at each iteration k the algorithm calculates:

$$\Delta f = f_k - f_{k-1} \tag{4}$$

$$h_\theta = \theta_k - \theta_{k-1} \tag{5}$$

$$\widehat{J}_k = \widehat{J}_{k-1} + (\Delta f - \widehat{J}_{k-1} h_\theta - \frac{\partial f_k}{\partial t} h_t) \tag{6}$$
$$(\lambda + h_\theta^T P_{k-1} h_\theta)^{-1} h_\theta^T P_{k-1}$$

$$P_k = \frac{1}{\lambda}(P_{k-1} - Pk - 1 h_\theta (\lambda + h_\theta^T P_{k-1} h_\theta))^{-1} h_\theta^T P_{k-1} \tag{7}$$

$$\theta_{k+1} = \theta_k - (\hat{J}_k^T \hat{J}_k)^{-1} \hat{J}_k^T (f_k + \frac{\partial f_k}{\partial t} h_t) \tag{8}$$

These group of equations represent a dynamic recursive least squares estimation to provide on-line estimates of the Jacobian ($\widehat{J}_k$) .The term h_t is a time increment and is defined as $h_t = t_k - t_{k-1}$; the term $\frac{\partial f_k h_t}{\partial t}$ predicts the change in the error function for the next iteration, where λ, $0 < \lambda \leq 1$, is the forgetting factor and represents the Image Jacobian in the k instant. It is important to remark that the image Jacobian referred in this section ($\widehat{J}_k$) is different to the Jacobian (J), that we defined in the previous section for the Cartesian controller.

4.3 Adaptive Neuro-Fuzzy Inference System (ANFIS)

As mentioned in [8], an ANFIS is a kind of adaptive network that acts as a framework for adaptive fuzzy inference systems. A fuzzy inference system is a popular computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. The ANFIS neural network is shown schematically in figure 3 and will be reviewed here briefly based in [8].

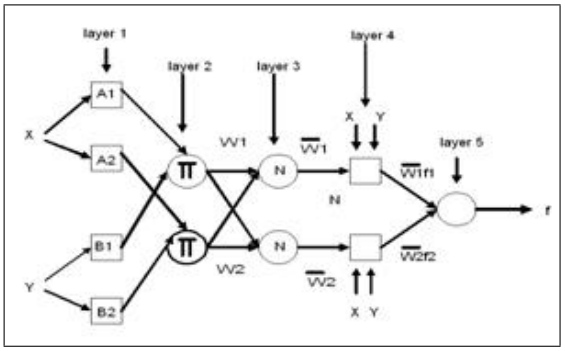


Figure 3: ANFIS architecture

For a first-order Sugeno fuzzy model, a typical rule set with two fuzzy if-then rules can be expressed as:

$$\begin{array}{llllllllll} Rule & 1: & if & x & is & A_1 & and & y & is & B_1 \\ Then & f_1 & = & p_1 x & + & q_1 y & + & r_1 & & \end{array}$$

$$\begin{array}{llllllllll} Rule & 2: & if & x & is & A_2 & and & y & is & B_2 \\ Then & f_2 & = & p_2x & + & q_2y & + & r_2 & & \end{array}$$

In figure 3 nodes of the same layer have similar functions, we will denote the output node i in layer l as $O_{i,j}$. Where x and y are referred to any input to the network, and can be any kind of data, they could be image features or atmospherical information, depends of the network application. Every node i in this layer is an adaptive node with a node output defined by:

$$\begin{array}{llll} O_{1,i} & = & \mu_{A_i}(x), & for \quad i = 1, 2, \quad or \\ O_{1,j} & = & \mu_{B_{i-2}}(y), & for \quad i = 3, 4, \end{array} \tag{9}$$

Where x (or y) is the input to the node, and A_i or B_{i-2} is a fuzzy set associated with this node. A_i and B_i can be any parameterized membership function. For example, A_i can be characterized by the generalized bell function:

$$\mu_{A_i}(x) = \frac{1}{1 + [(\frac{x - c_i}{a_i})^2]^{b_i}} \tag{10}$$

Where a_i, b_i, c_i are the bell function parameters. In the layer 2, every node in this layer is a fixed node which multiplies the incoming signals and outputs the product, for instance:

$$O_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1, 2 \tag{11}$$

In layer 3, the ith node calculates the ratio of ith rule's firing strength to the sum of all rules' firing strengths:

$$O_{3,i} = \bar{w_i} = \frac{w_i}{w_1 + w_2}, i = 1, 2 \tag{12}$$

In layer 4, every node in this layer is an adaptive node with a node function

$$O_{4,i} = \bar{w_i} f_i = w_i(p_i x + q_i y + r_i) \tag{13}$$

Layer 5, contains a single node which computes the overall output as the summation of all incoming signals:

$$O_{5,i} = overall \ \ output = \sum \underline{i} \bar{w_i} f_i = \frac{\sum_i w_i f_i}{\sum \underline{i} w_i} \tag{14}$$

Training the network consists of finding suitable parameters for layer 2 and layer 4, gradient descendent methods and least squares methods are used in the training phase.

4.4 Multilayer Perceptron (MLP)

Exist different kinds of neural networks and their classification is related to their learning algorithm or architecture, we will use a multi-layer perceptron neural network, MLP neural networks are classified as networks with a supervised learning algorithm. The MLP has an architecture based in 3 layers, an input layer, a hidden layer and an output layer. In [2] is stated that networks with just two layers of weights are capable of approximating any continuous functional mapping. Based on the work written by Terra in [20] will be described a MLP neural network, for one hidden layer, presenting in the sample n, the input vector $x_n = [x_{1n} x_{2n} \dots x_{pn}]^T$, for a p-dimensional input vector and a q-dimensional output vector, the MLP output of the neuron k (where $k = 1, 2, \dots, q$) is given by

$$\hat{y_k}(x_n) = \varphi_k(\sum w_{kj} \varphi_j(\sum w_{ji} x_{in})) \tag{15}$$

Where m is the number of neurons in the hidden layer, w_{kj} is the weight between the neuron j (hidden layer) and the neuron k (output layer), w_{ji} is the weight between the neuron i (input layer) and the neuron j (hidden layer), φ_k is the nonlinear activation function in the output layer and φ_j is the nonlinear activation function in the hidden layer.

4.5 Radial Basis Function (RBF)

These kind of neural networks compute the activation of a hidden unit calculating the distance between the input vector and a prototype vector as stated in [2], the training of these networks is faster than the multilayer perceptron, these networks have a two stage training procedure, in the first stage are calculated the parameters of the radial basis function that are related to the hidden neurons, this first stage uses an unsupervised training technique, in the second stage are calculated the final weights of the layer. The radial basis function approach

introduces a set of N basis functions, one for each point, that takes the form $\phi(\|x - x^n\|)$ where ϕ is some non-linear function, and depends of the Euclidean distance between x and x^n, where x and x_n are input vectors, the output of the mapping is then represented as a linear combination of the basis function:

$$h(x_n) = \sum w \sum w_n \phi(\| x - x^n \|) \tag{16}$$

and its matrix form is represented as:

$$\mathbf{\Phi w} = \mathbf{t} \tag{17}$$

t, represents the network targets, if exists the inverse matrix, we can obtain:

$$\mathbf{w} = \mathbf{\Phi^{-1} t} \tag{18}$$

There are many kinds of basis functions but the most common is the Gaussian:

$$\phi = exp(-\frac{x^2}{2\sigma^2}) \tag{19}$$

5 Motor Babbling

The process known as "motor babbling" in developmental robotics, is a process in which the robot manipulator moves randomly exploring its workspace. In order to construct the forward model are stored: the angular position of the robot joints, the Cartesian position of the robot end effector, and the image coordinates of the end effector position as seen by the visual system. A manipulator puma 560 was created in the matlab robotics toolbox and two fixed cameras are simulated in the Epipolar Geometry Toolbox. At the end of the motor babbling phase 2160 samples were collected. the reason to collect such big number of samples is to get enough data for a good model estimation, and these samples are distributed uniformly through the robot workspace. There are two simulated cameras one for the right eye and the other one for the left eye. The internal parameters for the two cameras are the same and are shown in Table 1. Both cameras have a rotation of -90 degrees with respect to x axis and are located at [0, -0.1, 0.8] and [0, 0.1, 0.8] in the world coordinates.

Table 1: Camera internal parameters

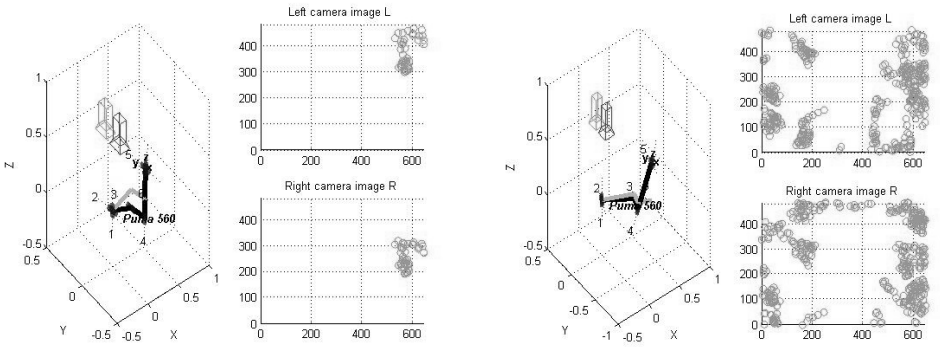
Parameter	Value
image pixel size	640×480
Orthogonality factor of the CCD image axes	1
Number of pixels per unit distance in image coordinate times the focal length	200

The robot workspace was babbled in 36 stages. At the beginning of each stage the robot joint 0 is set to 10n degrees, where n is the stage number. Each stage is made of 60 babbling steps, in each of these babbling steps the robot moves every joint randomly and increasingly in order to explore its workspace. Figure 4(a) shows the robot and the resulting image points at the beginning of the babbling process. Figure 4(b) shows the end of the 36 babbling stages and the resulting images of the end effector points in both cameras.

6 Forward Model Creation

6.1 Forward Model Creation using ANFIS

The forward model was constructed using the data stored in the babbling motor phase, for the construction of the forward model we used the ANFIS neural networks from the fuzzy logic matlab toolbox. Figure 5 shows how the forward model is created using the ANFIS toolbox of Matlab. The input data for the neural network comes from the positions of the end effector in the image field of view and the outputs are the coordinates (x,y,z) in Cartesian space of the end effector, for calculating this position is used the forward kinematics of the robot using the joint angles of the manipulator stored in the babbling motor phase. Three ANFIS neural networks were constructed, one for each Cartesian coordinate (x,y,z). Every network has 4 inputs that are the image feature coordinates (u_L, v_L, u_R, v_R) , and one output that is one of the coordinates of the end effector Cartesian position (p_x, p_y, p_z). Table 2 shows the number of rules that are created using a subclustering radius of 0.5., for the three created



(a) (b)

Figure 4: Babbling motor. (a) The left subplot shows the position of the robot after the second babbling stage (joint 0 is initialized to 20 degrees). The top and bottom subplots show the imaged end effector points at each babbling step. (b) The left subplot shows the position of the robot at the end of the babbling. The right top and right bottom subplots are the end effector accumulated positions of the end effector in the right and left camera.

Table 2: ANFIS Initialization

Neural Network	Subclustering radius	Number of rules
ANFIS 1(x)	0.5	6
ANFIS 2(y)	0.5	8
ANFIS 3(z)	0.5	10

neural networks. Analytical methods could be used also to construct the model proposed in this work, but they need a calibration step and know the internal and external camera parameters, the process of constructing that model could be prone to error if the parameters are not well estimated, besides a distortion correction phase is necessary once the model has been constructed.

6.2 Forward Model Creation using MLP and RBF

Using the same data stored in the babbling motor phase it was constructed the forward model employing multilayer perceptron neural networks. So, three networks were created with the same inputs and outputs that were used for the ANFIS networks, before the training phase the data was preprocessed in the intervals of 1 and -1, 10 neurons were used in the hidden layer of every network; the number of epochs during the training phase of every network were 150, 33, 69 respectively.

Furthermore, the data created in the babbling motor phase was also used for constructing the radial basis functions networks, after preprocessing the data 3 neural networks were created with the following training parameters , sum squared error set as 0.06 for the three networks, and spread constant defined as 0.75, 0.9 and 0.6 respectively for the three RBF neural networks.

7 Reaching Following a Moving Target

7.1 Piepmeir's Algorithm for Following a Moving Target

Initially, the robot is in the position defined in joint coordinates as [-1.4, -0.3, -1.47, 0, 0, 0], the final objective is that the robot could follow a simulated moving target defined by the equations:

$$\begin{aligned} x &= x_0 + 0.2 * sin(wt) \\ y &= y_0 + 0.1 * cos(wt) \end{aligned} \tag{20}$$

The initial position of the target in Cartesian space is with $w = 0.01$, $x_0 = 0.25$, $y_0 = 0.2$ and $z_0 = 0.4$. For the simulations, the camera frame rate for this simulation is fixed at 1/60 sec. The simulation time step is 0.005. The

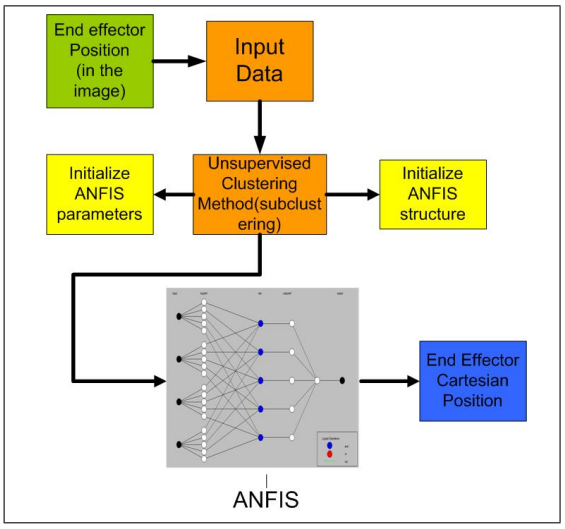
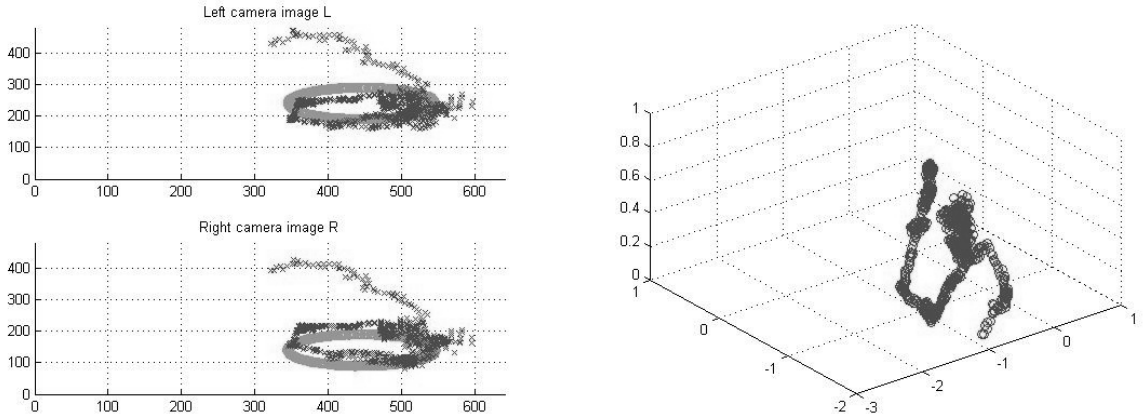


Figure 5: Construction of the forward model using an ANFIS based methodology.

number of iterations for the control loop is equal to 800 control steps (iterations).

Figure 6(a), shows the trajectory of the moving target (continuous line) and the trajectory followed by the end effector (cross marks) in the cameras image field of view; furthermore, figure 6(b) shows the trajectory of the end effector traversed in the 3D space. Figure 7, shows the error of the visual servoing controller following the moving target. This error is a normalized error that estimates in each iteration the difference between the desired image position and the actual position.



(a) (b)

Figure 6: Figure 6. End effector trajectories using Piepmeir's algorithm. (a) Trajectories followed by the target (green) and robot end effector (red) in the image field of view of the two cameras using Piepmeir's algorithm. (b) 3D Trajectory of the end effector using the Piepmeir's algorithm.

7.2 Proposed Architecture for Following a Moving Target Using ANFIS

The final objective of the robot is that it could follow a moving target trajectory on its workspace. So, first the robot employs the forward model encoded in the ANFIS neural networks to calculate the Cartesian position of the moving target, this position is used as the desired position to be reached for the Cartesian controller. In the experiments the initial position of the robot is also [-1.4, -0.3, -1.47, 0, 0, 0] in joint coordinates. The joint

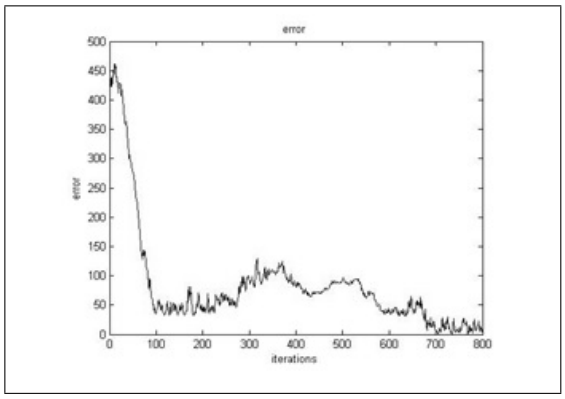
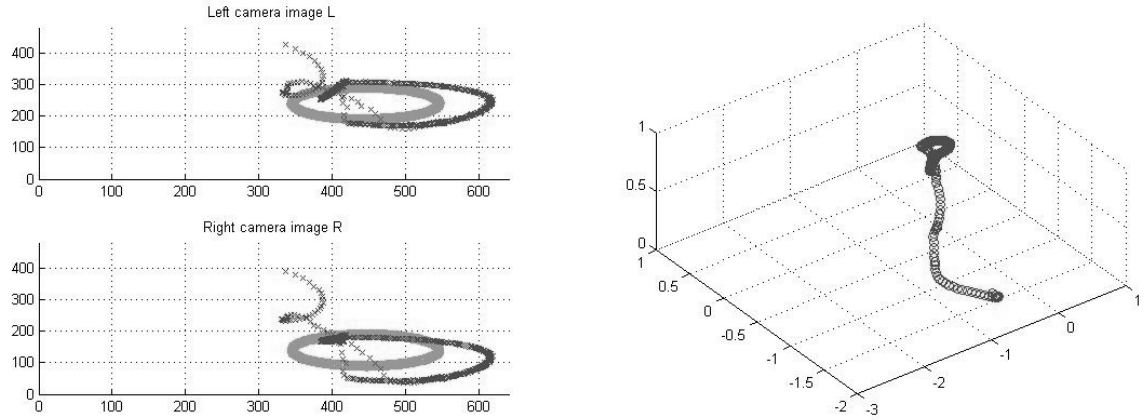


Figure 7: Visual servoing controller error for following a moving target.



(a) (b)

Figure 8: End effector trajectories using ANFIS. (a) Trajectories followed by the target (green) and robot end effector (red) in the image field of view of the two cameras using the proposed architecture based in forward models and a Cartesian controller. (b) 3D Trajectory of the end effector using the proposed architecture based on forward models.

velocities fixed in the experiments are six times the velocities used with the Piepmeier's approach, we tried the same velocities with the Piepemeier's algorithm, but just doubling the velocities in the joints will generate worse results. Figure 8(a), shows the trajectories followed in the image field of view for the target (continuous line) and the end effector position (cross marks). Figure 8(b) shows the traversed trajectory of the end effector in three dimensional space. Figure 9 shows the error obtained with the proposed architecture. This error in image space is the same defined for the experiments with the Piepmeiers's approach.

7.3 Proposed Architecture for Following a Moving Target Using MLP and RBF

Figure 10 shows the results of the proposed architecture in this paper using MLP networks, figure 10(a) shows the image trajectory of the end effector and figure 10(b) shows the error obtained with the controller.

Furtheremore, Figure 11, shows the results using forward models with RBF networks, figure 11(a) shows the image trajectory of the end effector and figure 11(b) shows the error obtained with the controller.

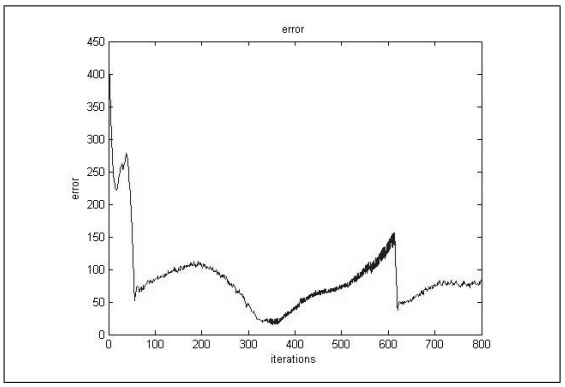
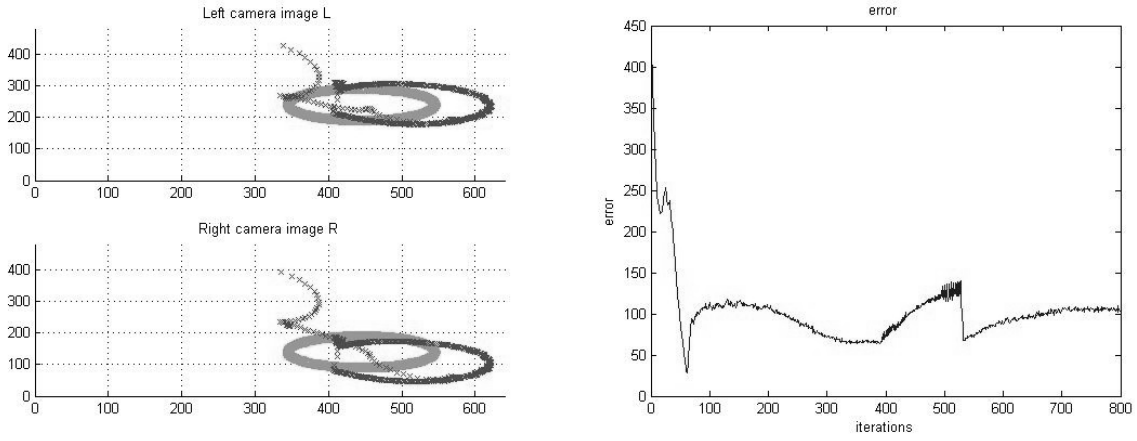


Figure 9: Proposed architecture error for the task of following a moving target using ANFIS neural networks.



(a) (b)

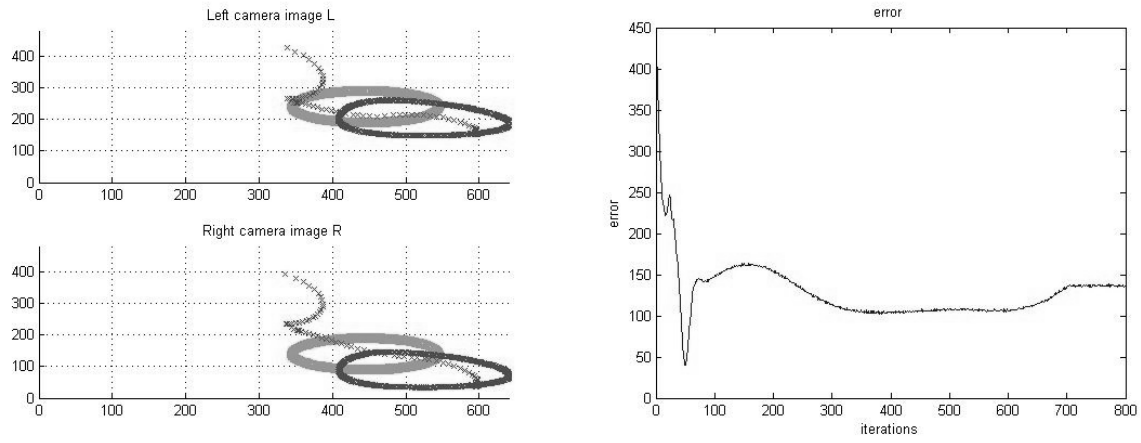
Figure 10: Following a moving target using MLP. (a) Trajectories followed by the target (green) and robot end effector (red) in the image field of view of the two cameras using the proposed architecture based in forward models and a Cartesian controller. (b) Proposed architecture error for the task of following a moving target using MLP neural networks.

7.4 Discussion

Based on the results presented in the paper, we could emphasize that the proposed architecture succeeds in its initial purpose of following a moving target, the trained ANFIS neural networks embedded a visual information map, the map was used in the task of tracking a target posteriorly. In order to compare the potential of the proposed architecture, results obtained using the Piepmeier's approach for following a moving target are also shown.

Analyzing both methods, it is possible to observe that the proposed architecture has better results in the Cartesian space, The $3D$ trajectory followed by Piepmeir's algorithm is not smooth compared to the $3D$ trajectory generated by the proposed solution in this paper, this can be noticed clearly in figure 8(b), where it is shown the end effector tracking a target with a regular circular trajectory in Cartesian Space. In the image space the task of tracking the target is accomplished as well with our approach, but the Peipmeir's approach is closer to the target compered with ours as it is concluded from the errors in the image space generated for the tested algorithms. However, our algorithm generates a smoother trajectory in image space compared with Piepmeir's algorithm.

ANFIS neural networks have a very powerful performance in tasks of regression and classification, and are used in different fields of engineering and machine learning, that is the reason for its selection in the architecture developed in this work.In order to validate our results two traditional neural networks (MLP and RBF) have



(a) (b)

Figure 11: Following a moving target using RBF. (a) Trajectories followed by the target (green) and robot end effector (red) in the image field of view of the two cameras using the proposed architecture based in forward models and a Cartesian controller. (b) Proposed architecture error for the task of following a moving target using RBF neural networks.

also been tested. So, we repeated the experiments using forward models embedded in RBF and MLP neural networks and the results are very similar to the ANFIS performance, just RBF had a little better performance in image space compared with the other networks. Future applications of the control architecture proposed could be visualized as a controller for a manipulator used in rehabilitation robotics that could be used for patients that suffered a stroke in order to recover their motor capabilities guided by the robot; another application could be visualized as a manipulator for a welding task in order to track a trajectory.

8 Conclusion

The main discernible contributions proposed by the paper are a new application of a forward model for following a moving target task using a Cartesian controller with image processing feedback with a robot manipulator. Another paper contribution is the proposal of a new control architecture for the task of tracking a moving target, an architecture that is founded in the developmental robotics field using past information embedded in ANFIS neural networks obtained during a babbling motor phase to accomplish the task of following a target, our architecture is able to generalize a new information based in past experiences ("motor babbling") and is more biological plausible. In order to show the applicability of our approach we also constructed forward models using MLP and RBF neural networks, obtaining very similar results to ANFIS networks. Finally, our last contribution is the comparison of two approaches for solving the problem of tracking a moving target with a robot manipulator; as a result of the comparison, we could state that we have a better performance in the Cartesian space and an acceptable performance in the image space for the tracking with the approach proposed in this paper compared with the Piepmeier's approach.

Acknowledgements

We would like to thank to the reviewers for their comments that helped to improve the paper.

References

[1] B.M. Becerra, S. Cook, and J. Deng. Predicted computer torque control of a puma 560 manipulator robot. In *IFAC World Congress-Internantional Federation of Automatic Control*, 2005.

[2] C. Bishop. Neural networks for pattern recognition. *Evolutionary Computation*, 10:99–127, 2002.

[3] P.I. Corke. A robotic toolbox for matlab. *IEEE in Robotics and Automation Magazine*, 3:24–32, 1996.

[4] Jhon J. Craig. *Introduction to Robotics, Mechatronics and Control*. Pearson Prentice Hall.

[5] D.F.T. Gamarra. Forward models with cluster validity criteria applied in ballistic reaching for visual servoing. *International Review on Modelling and simulations*, 6:1939–1947, 2013.

[6] D.F.T. Gamarra, L.K.Pinpin, C. Laschi, and P. Dario. Forward models applied in visual servoing for a reaching task in the icub humanoid robot. *Applied Bionics and Biomechanics*, 6:345–354, 2009.

[7] S. Hutchinson, E.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Auromation*, 12:651–670, 1996.

[8] J.S.R. Jang and C.T. Sun. Neuro-fuzzy modeling and control. *Proc. of the IEEE*, 83:378–406, 1995.

[9] J.A.Piepmeier, G.V. McMurray, and H. Linkin. A dynamic jacobian estimation method for uncalibrated visual servoing. In *Proc. of the IEEE International Conference on Advance Mechatronics (ASME)*, pages 944–949, 1999.

[10] J.A.Piepmeier, G.V. McMurray, A. Pfeiffer, and H. Linkin. Uncalibrated target tracking with obstacle avoidance. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1670–1675, 1999.

[11] M. Kawato. Internal models for motor control and trajectory planning. *Trends in Cognitive Science*, 9:718–727, 1999.

[12] L.K.Pinpin, D.F.T. Gamarra, C. Laschi, and P. Dario. Forward model creation for visual servoing in a six link manipulator. In *Proc. of the IEEE Conference on Biomedical Robotics and Biomechatronics (BIOROB)*, 2008.

[13] G.L. Mariottini and D. Prattichizzo. A robotic toolbox for matlab. *EGT for multiple view and visual servoing: robotics vision with pinhole and panoramic cameras*, 12:26–39, 2005.

[14] L. Natale, F. Nori, G. Metta, M. Fumagaldi, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz, and L. Sandini. *Intrinsically motivated learning in natural and artificial systems*, chapter The iCub platform a tool for studying intrinsically motivated learning. Baldassarre, Gianluca and Miroli, 2012.

[15] F. Nori, G. Metta, and L. Sandini. *Robust Intelligent Systems*, chapter Exploiting motor modules in modular context. Alfons Schuster, 2008.

[16] R. Saegusa, G. Metta, and L. Sandini. Body definition based on visuomotor correlation. *IEEE Transactions on Industrial Engineering*, 59.

[17] P. Sanchez-Sanchez and F. Reyes-Cortes. A new cartesian controller for robot manipulators. In *Proc. of the IEEE International Conference on Intelligent Robotics and Systems (IROS)*, pages 2059–2064, 2005.

[18] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics, Modelling, Planning and Control*. Springer, London-UK, 2009.

[19] E. Tellen, D. Corbetta, K. Kamm, J.P. Spencer, K. Schneider, and R.F. Zernicke. The transition to reaching: Mapping intention and intrinsic dynamics. *Child Development*, 64:1058–1098, 1993.

[20] M.H. Terra and R. Tinos. Fault detection and isolation of robotic systems using a multilayer perceptron and a radial basis function. In *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1880–1885, 1998.

[21] C. von Hofsten. Eye-hand coordination in the newborn. *Developmental Psychology*, 18:450–461, 1982.

[22] J.A. Walter and K.J. Shulten. Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Transactions on Neural Networks*, 59:86–95, 1993.

[23] D. M. Wolpert, R.C. Miall, and M. Kawato. Internal models in cerebellum. *Trends in Cognitive Science*, 2:338–347, 1998.