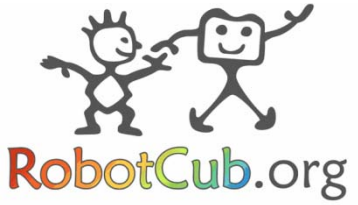

Robotic Open-architecture Technology for
Cognition, Understanding and Behavior

Project No. 004370

RobotCub
Development of a Cognitive Humonoid Cub

Instrument: Integrated Project
Thematic Priority: Cognitive Systems

D 3.5 Robotic Implementation of models of sensory-motor coordination for reaching tasks

Due date:
Submission Date:

Start date of Project: **01/09/2004** Duration: 60 months

Organization name of lead contractor for this deliverable: **SSSA**

Responsible Person: **Cecilia Laschi, Paolo Dario**
Authors: **Cecilia Laschi, Fernando Gamarra, Nicola Greggio.**

Revision: **4.0**

Project co-funded by the European Commission within the Sixth Framework Program (2002-2006)		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other program participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Abstract

Human infants learn how to develop sensory-motor coordination in their first years of life. Sensory-motor coordination involves different stages. Considering the use of a hand for tasks as reaching and grasping, the infant develops his capabilities by achieving different levels of motor control. First, he has to learn how to control his own arm in order to approach the hand close the object. Then, he has to grasp the object. When we use our digits to manipulate objects the applied fingertip forces and torques tangential to the grip surfaces are a result of complex muscle activity.

In this work we investigate the generation of internal models for reaching and grasping. We use bio-inspired techniques, to make a robot able to develop capabilities to reach and grasp objects itself, as human beings do. We believe that taking inspiration from the human system may help in developing a highly capable robotic controller, this robot being in turn an interesting platform for neuroscientists to test hypotheses on movement generation. As it was demonstrated in previous works [1] and [2].

In order to develop this architecture we implemented different modules. The sensory motor map is condensed in a forward model of the kinematics of the robot. The forward model is found through a neural network that exploits information that comes from the clustering topology in a data distribution. The neural network used is an Adaptive Neuro-Fuzzy Inference System (ANFIS).

The results obtained for the calculation of a forward model, the vision system and the servocontroller module as part of our architecture for the reaching are described.

1.-Introduction

In this work we implemented a model that makes the humanoid platform iCub able to self-compute its internal models in order to perform a correct reaching of objects with its arms.
In our architecture the robot uses the vision module in order to detect the position of its end-effector (i.e. its hand). This, together with the arm encoder information, will be the input of the neural network.
The general scheme of our architecture is described in Figure 1. This architecture involves different modules to be developed. The first module is the forward model that has been calculated using a neural network.
Once the forward model is calculated, the model is used to estimate an initial Jacobian that is necessary to implement for the servo controller. The servo controller needs a vision system that will give the feedback necessary while the arm is approaching to the target. The servocontroller will send the commands that are necessary to move the joints of the arm to accomplish the task.

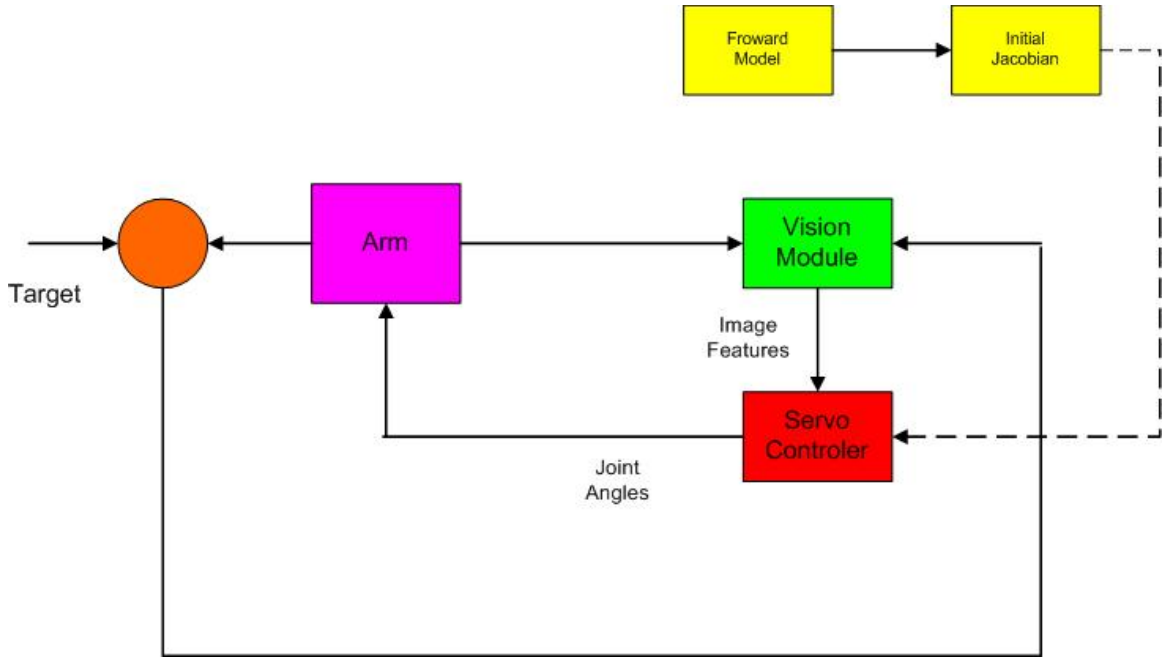

Figure 1. General Platform for the reaching based on a forward model

2.-Forward Model Module

This deliverable presents the results in simulation to obtain the forward model in the robot arm. The forward model will be obtained after a phase of babbling in which we captured joint and end-effector information. The end-effector information comes from the vision system. This information is processed for the ANFIS neural network in order to obtain the Arm forward model.

In Figure 2 we can see a general scheme for the motor babbling phase. It has been used the ANFIS as the neural network for the simulation. The ANFIS is initialized with a clustering algorithm. The ANFIS is trained with data that comes from a simulated robot. In the next sections are described the mathematical tools, procedure and results obtained to calculate a forward model of a robotic arm in simulation.

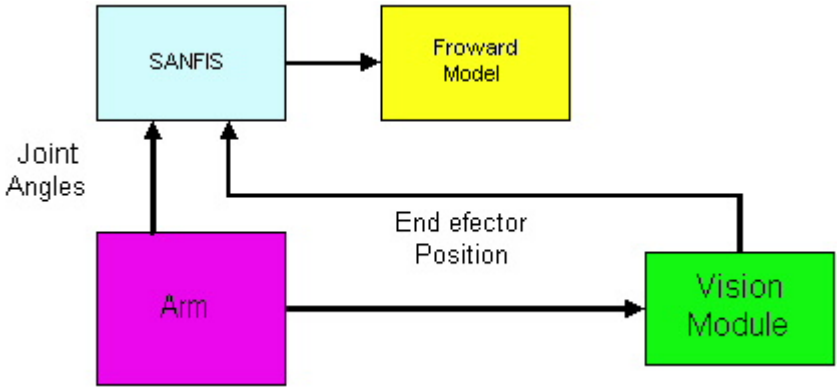

Figure 2. Calculus of the Forward model (sensory motor map) using the ANFIS

3.-Adaptive Neuro Fuzzy Inference System (ANFIS)

We review briefly the ANFIS [3] that is the neural network that we employed to calculate the forward model. It is used a two input example. Adaptive Neuro-Fuzzy Inference Systems are Fuzzy Sugeno models put in the framework of adaptive systems, a fuzzy Sugeno type is composed by rules of the type:

Rule 1: if x1 is A1 and x2 is B1,
then f1 = a1x1+b1x1+c1

Rule 2: if x1 is A2 and x2 is B2,
then f2 = a2x1+b2x2+c2

The architecture of the network is illustrated in figure 3. In the first layer the degree of the membership of the input is computed using as a membership function a Gaussian:

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

where a_i, b_i and c_i are the parameters of the Gaussian function.

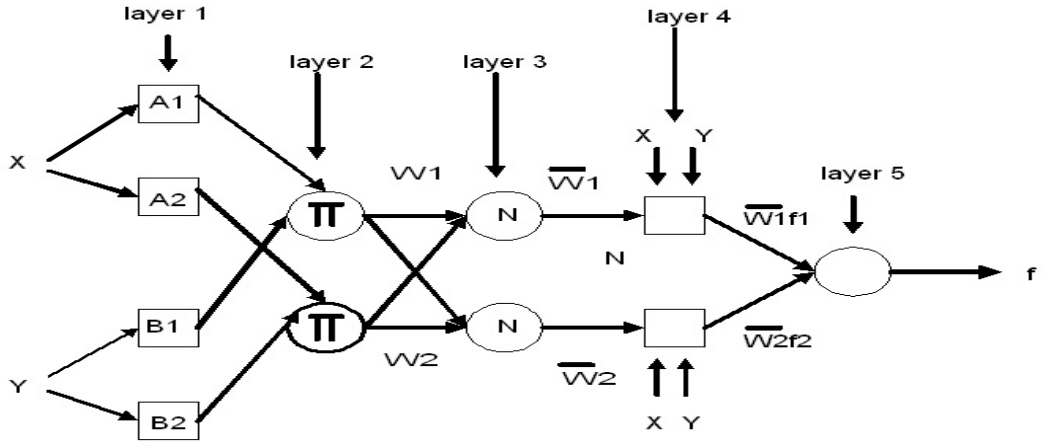

Figure 3.ANFIS architecture

The second layer calculates the firing strength (or weight) w_i of the -ith rule:

$$w_i = \mu_{A_i}(x_1)\mu_{B_i}(x_2)$$

In the third layer the firing strengths are normalized with the sum of all rule's firing strengths:

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}$$

In the fourth layer the output is calculated as the product of the normalized firing rate and the parameters set:

$$\overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i)$$

Finally, in the fifth layer is calculated the overall output as the addition of all incoming

signals,

$$\sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

For the training of the network to make the output match the desired values, different methods as the gradient descendent rule, the least square error or an algorithm that could be a hybrid of the two methods can be applied.

4.- Forward Model Creation

The data collected from the babbling phase is used to create a forward model of the robot. Figure 4 shows how the forward model is constructed using the ANFIS toolbox of Matlab. The input data is a set that includes the end effector position in the image and joint angles of the manipulator. The input data is clusterized using the unsupervised clustering algorithm of the toolbox that uses the subclustering algorithm [4]. The unsupervised clustering algorithm gives the initial estructure of the network (number of fuzzy rules and parameters for the initialization of the membership functions).

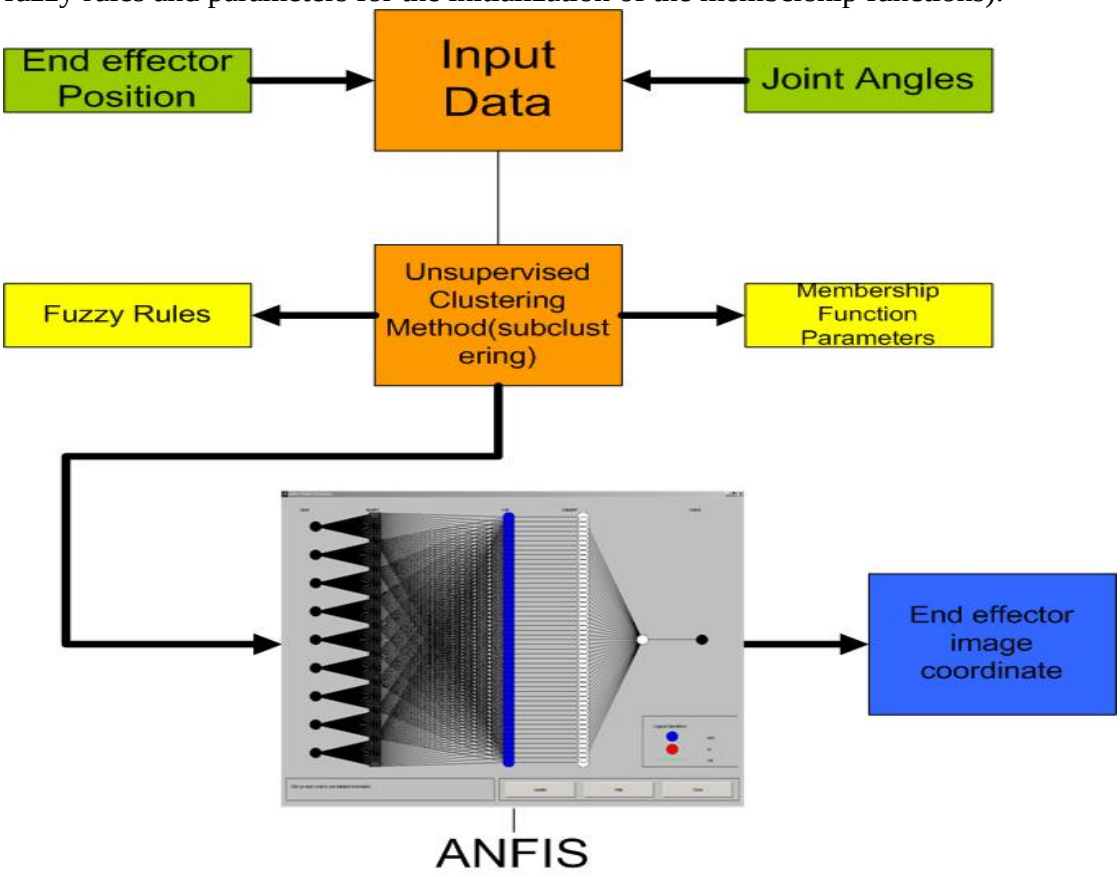

Figure 4 Construction of the forward model using an ANFIS-based methodology

A total of four ANFIS neural networks have been constructed – one ANFIS for each image feature coordinate (u_L,v_L,u_R,v_R). Each neural network has 9 inputs (q_0, q_1, q_2, q_3, q_4, q_5, p_x, p_y, p_z). The first 6 inputs are the angular positions of the joints of the manipulator and the other 3 inputs are the coordinates of an end-effector point (with respect to the end-effector local frame). The output of the network is an image coordinate (u or v) of the end-effector position (p_x, p_y, p_z) in one of the cameras (left or right). A total of five feature points have been tracked because it has been seen in the simulations that as the number of tracked points is increased the robustness of the algorithm grows.

5.-Vision Module

The vision module receives the images from the two cameras mounted on the iCub head. It is responsible of processing these images in order to obtain the relevant information about the object to be grasped. These are: shape, dimension, orientation, and position within the 3D surrounding environment (this is accomplished by triangulating the information received from the binocular vision and the head and neck encoders). In our particular case we made our experiments by using a ball of different colors as object of interest.

In order to detect the ball, and all its features, we implemented a simple but efficient image processing algorithm. We detect the ball by means of a color filter. The pixels of the ball are detected by setting color thresholds for the pixels belonging to the ball. We implemented a technique that creates a database for all the possible colors. Each color (detected with an image of interest) is represented with the HSV representation by its histogram evaluated within the image it owns to. Then, our application for the iCub loads the correspondent color representation from this database at runtime.

Once the ball pixels are identified, the image is converted into a binary image with ball pixels set to '1'. The binary image contains not only the blob relative to the ball, but also other smaller blobs caused by color variation in the image. For the identification of the blob corresponding to the ball, we use a connected components algorithm. We assume the largest blob is the ball, so we look for the blob with the largest area. Subsequently we proceeded by applying the algorithm by Maini [5], to the found blob, in order to detect all the parameters of the curve that describes the boundary of the blob. This is a new interesting LS technique, the Enhanced Least-Square Fitting of Ellipses (EDFE), that has been developed recently, and it was proposed in [5]. It is a LS procedure that improves the work described in [6]. In this work, Fitzgibbon et al. developed a direct computational method (i.e. B2AC) based on the algebraic distance with a quadratic constrain. This new approach overcomes the state of the art by solving the problems of numerical instability that can produce completely wrong results, such as infinite or complex solutions, not reported in the original work [6].

We tested our algorithms by using the icub simulator. We used the iCub ODE simulator present in the iCub repository. Moreover, we slightly modified the simulator in order to create different scenarios for our experiments (such as by changing the color of the ball, by removing the table, etc.). In Fig. 5a and 5b an example of the ball detection algorithm output is shown. In Fig. 5a the input to the left camera is presented, i.e. the experimental scenario, while in Fig. 5a output of the algorithm is presented, These images are the input image as seen by the robot with the egocentric view (5a) and the same image with the superimposition of an ellipse, drawn by using the characteristic parameters obtained by computing the EDFE.

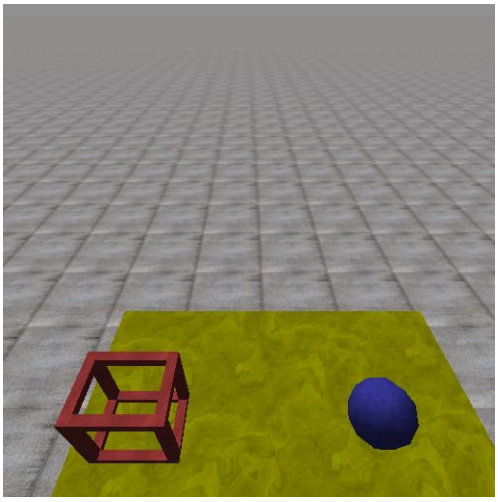

Fig. 5a

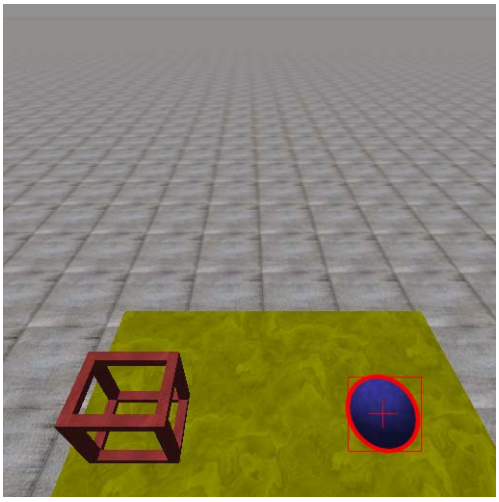

Fig. 5b

Fig. 5 The input image, as seen by the robot with the egocentric view (2a) and the same image with the superimposition of an ellipse, drawn by using the characteristic parameters obtained by computing the EDFE.

In addition, we implemented a tracking algorithm that directly commands the head of the robot, in order to be able to reconstruct the target object position (in terms of its centroid) by triangulating the information of the neck and head encoders (see Fig. 6).

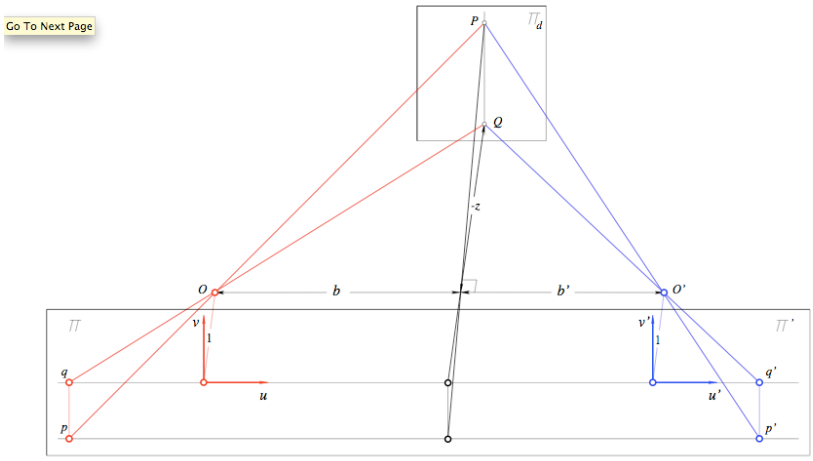

Fig. 6 An example of triangulation of an object. Once the object has been detected by the cameras and tracked, it is possible to evaluate its 3D position in the surrounding space geometrically, by knowing the encoders position of the head and the neck of the robotic platform (in this case we refer to a humanoid robot).

This will be fundamental for computing the Sensory-Motor maps, as will be explained in the next section In Fig. 7 a print screen is depicted, that shows an operative situation in which the simulator tracked the ball. The iCub program we implemented the position of the ball (which is the target to be grasped in this case), in terms of cartesian position. We adopted the same system reference as the simulator, in order to be fully compatible with the measures and the signs adopted in the virtual environment[1]. This allowed us not only an easier implementation of the software, but also to test easily our tracking algorithm by simulating different scenarios, i.e. by putting the ball in different positions (under the table, as in Fig. 5 or on the floor, as in Fig. 7).

[1] The reference system is centered on the floor plane, at the center of the pole that sustains the robot. The x axis evolves along the front of the robot, the y axis runs along the left of the robot, and the z axis evolves along its height.

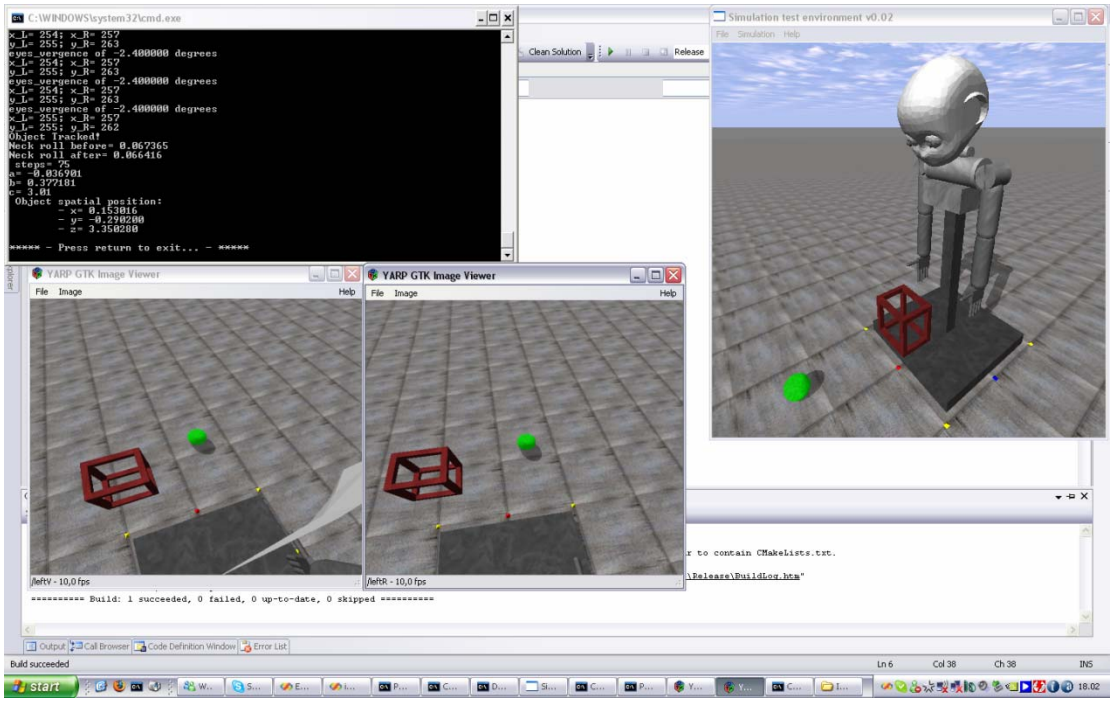

Fig. 7 A screenshot depicting the moment in which the simulated robot tracked the position of the ball in the 3D surrounding environment. Therefore, our program uses the encoders information to triangulate the position of the centroid of the object within the simulated space.

Again, this is a test for using this software for testing our algorithms for the sensory-motor maps generation. In fact, with the simulator it is possible to test our neural networks that generate the internal models for the sensory-motor maps without having the hardware iCub platform in the laboratory. Clearly, the simulator information is not exhaustive, but it is a good approximation for the software debug before using it on the rear robot, which can be extremely dangerous in case of wrong movements, due to the elevate torque of its motors.

6.-The Servocontroler Module

In order to accomplish successfully the reaching task it is followed a classical servocontroller approach in which we have defined as a control law:

$$\tilde{\theta}(i) = \tilde{\theta}(i-1) + \tilde{J}\#\alpha(i)(x_{target} - \tilde{x}(i-1))$$

The angle theta represents the joint angles of the robot. The new theta joint estimated angle is obtained from the previous estimated theta angle plus the inverse Jacobian estimation. The error is derived from the end-effector estimated position in the image and

the target position in the image.

In figure 8 is shown the initial position of the robot and the target. The other subplots of this figure display the representation of these points in the two cameras at the beginning of the reaching. Figure 9 shows the position of the robot when it has reached the target. Finally figure 10 shows the error obtained with the servocontroller and the convergence of this error. These results were obtained using the robotics toolbox for Matlab of Corke [7].

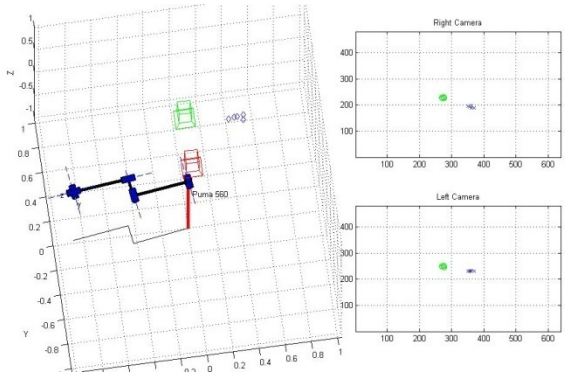

Figure 8. The left subplot has the position of the robot at the beginning of the reaching. The target position is represented by the blue points in the left subplot (feature points of the end effector at the end of reaching). The right top and right bottom subplots show the end-effector position (green circles) and the target image coordinates (blue crosses).

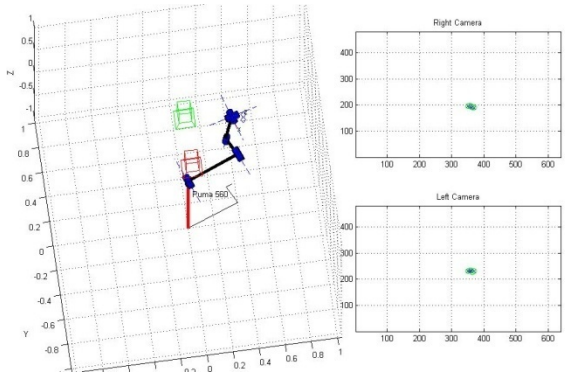

Figure 9. The left subplot has the position of the robot at the end of the reaching. The target is represented by the blue points in the left subplot (partially obscured by the end effector). The right top and right bottom subplots show the end-effector position (green circles) and the target (blue crosses). Since the robot end-effector has reached the target the blue crosses are overlapping with green circles.

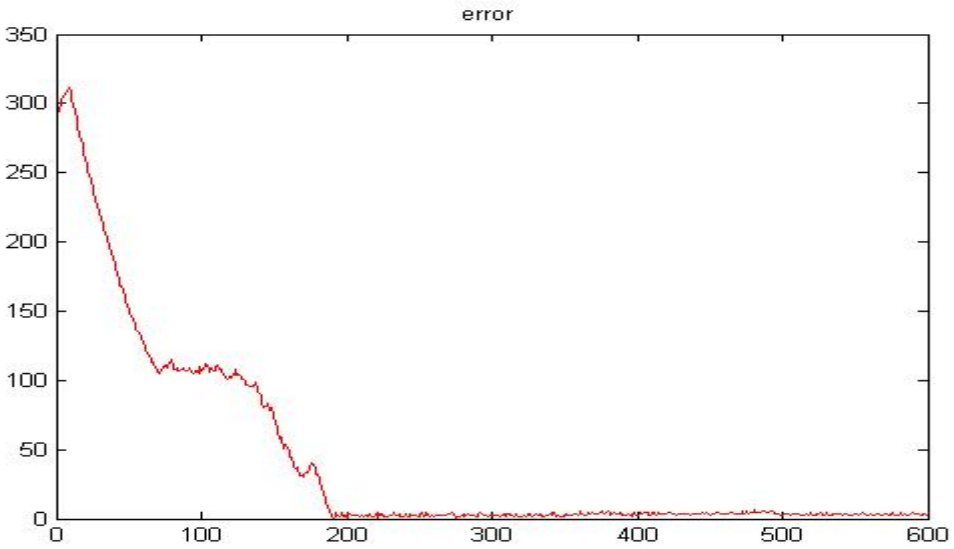

Figure 10. Error obtained with the servocontroller.

7.-Discussion

We have demonstrated in this deliverable how through a babbling motor phase similar to a self exploratory "babbling body" process developed by infants a forward model is constructed. The forward model is constructed using ANFIS neural networks. The forward model created serves to initialize optimally the image Jacobian that is used in the image-based visual servoing controller for a reaching task.

We have used for the first time in a robotics application a new algorithm: The EDFE for image processing, and a new way in which a forward model can be used. The forward model calculated for the ANFIS network and trained with data derived after a babbling phase in the 6 link robotic manipulator. With reference to the vision module we implemented the EDFE pattern recognition algorithm, which allows more precision in the recognition of object that can be assumed as particular cases of elliptical sections.

The method described here to find a forward model can be used with any robotic manipulator. The method through neural networks finds a relation between the joints of the manipulator and the position of the end-effector in the image space (image Jacobian).

We are currently working to optimize the trajectories of the end-effector using the forward model. The forward model is used in biological systems as a predictor as is stated by [8] and [9]. This characteristic is being used in our future work to deal with constraints as obstacle avoidance during the trajectory or some singularities due to the redundancy.

The forward model module and servocontroller module are being translated to the YARP architecture. We are integrating these different modules some of them developed in Matlab and others in the iCub ODE simulator to make all of them run in the YARP platform.

Bibliography

[1] G.Asuni, G. Teti, C. Laschi, E. Guglielmelli, P. Dario. “Extension to End-effector Position and Orientation Control of a Learning-based Neurocontroller for a Humanoid Arm” *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and systems*, October 9-15, 2006, Beijing, China

. [2] G. Asuni, G. Teti, C. Laschi, E. Guglielmelli, and P. Dario. “A bioinspired sensory-motor neural model for a neuro-robotic manipulation platform.“ *In 12th International Conference on Advanced Robotics (ICAR)*.Seattle, Washington, USA., Jul 2005.

[3] J. S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *Systems, Man and Cybernetics*, IEEE Transactions on, vol. 23, pp. 665-685, 1993.

[4] Yager, R. and D. Filev, "Generation of Fuzzy Rules by Mountain Clustering,". Journal of Intelligent and Fuzzy Systems, vol 2, No 3,pp 209-219, 1994.

[5] E.S. Maini, “Enhanced direct least square fitting of ellipses”, IJPRAL, vol 20, no 6, pp. 939-954, 2006.

[6] A. Fitzggibbson, M. Pilu, R. Fisher, “Direct least square fitting of ellipses, IEEE Trans. PAMI 21(1999)476-480.

[7] Corke Peter, "A Robotics Toolbox for MATLAB" IEEE Robotics and Automation Magazine, Magazine, vol 3. pp 24-32, 1996.

[8] D. M. Wolpert, R. C. Miall, and M. Kawato, "Internal models in the cerebellum,". *Trends in Cognitive Sciences*, vol. 2, pp. 338-347, September 1998.

[9] M. Kawato, "Internal models for motor control and trajectory planning,". Current Opinion Neurobiology vol. 9, pp. 718-27, Dec 1999.