

Forward models applied in visual servoing for a reaching task in the iCub humanoid robot

Daniel Fernando Tello Gamarra*, Lord Kenneth Pinpin, Cecilia Laschi and Paolo Dario
*Scuola Superiore Sant'Anna, ARTS Lab (Advanced Robotics Technology and Systems Laboratory),
V.le Rinaldo Piaggio, 34-56025 Pontedera (PI), Italy*

(Received 5 January 2009; final version received 26 May 2009)

This paper details the application of a forward model to improve a reaching task. The reaching task must be accomplished by a humanoid robot with 53 degrees of freedom (d.o.f.) and a stereo-vision system. We have explored via simulations a new way of constructing and utilizing a forward model that encodes eye–hand relationships. We constructed a forward model using the data obtained from only a single reaching attempt. ANFIS neural networks are used to construct the forward model, but the forward model is updated online with new information that comes from each reaching attempt. Using the obtained forward model, an initial image Jacobian is estimated and is used with a visual servoing controller. Simulation results demonstrate that errors are lower when the initial image Jacobian is derived from the forward model. This paper is one of the few attempts at applying visual servoing in a complete humanoid robot.

Keywords: forward models; ANFIS; robotics; visual servoing; neural networks

1. Introduction

Recent studies point to the possibility that the human brain could create internal models (Wolpert et al. 1998). Kawato (1999) defined internal models as neural mechanisms that can mimic the input/output characteristics, or their inverses of the motor apparatus. The internal models could be forward models or inverse models. Forward models can predict sensory consequences from efference copies of issued motor commands. Inverse models calculate the necessary feed forward motor commands from the desired trajectory information.

The process of creating forward models starts in infants when they are born. The newborn, through a self-exploratory phase of his kinematics and sensory feedback ('body babbling'), creates an internal model of his own kinematics and sensory system as described in Rao et al. (2004). Roboticians, looking at biology and specifically human development as a source of inspiration, have begun to use forward models in robots. For instance, in Sun and Scasellati (2005), a forward model that represents the forward kinematics of a manipulator was created using radial basis function neural networks. From the forward model, they analytically derived the robot Jacobian that is used in a control law that governs the reaching task. In Dearden and Demiris (2005) a mobile robot, after a babbling motor phase, learns a forward model based on a Bayesian neural network. The forward model was used by the robot in imitating human movements. In addition, in Sturn et al.

(2008) a forward model of a robot manipulator is learned by using a Bayesian network, after a babbling phase using a monocular camera.

Following a developmental robotics roadmap, this work tries to shed some light on the use of forward models for the visual servoing method that we intend to use for vision-based reaching. The forward model created is used to estimate an initial image Jacobian that becomes an important factor to determine the maximum performance attainable in reaching a task using a well-known visual servo controller.

Compared with Sun and Scasellati (2005), our method also uses a Jacobian derived from a forward model. However, instead of being derived analytically from a static forward model obtained via off-line training, our approach updates the forward model before each reaching attempt and consequently perturbs the updated forward model in order to obtain the image Jacobian. This strategy seems more biologically plausible because it uses past information from previous reaching attempts while avoiding analytical derivations of the image Jacobian. Since the forward model is updated online, extensive motor babbling and off-line training is avoided. An adaptive neural network-based fuzzy inference system (ANFIS) have been used to construct the forward models and the paper shows that they are able to represent with fewer rules the visuo-motor map and train very quickly.

The remainder of the paper is as follows: In the second section are described the algorithms used in this work;

*Corresponding author: Email: fgamarra@sssup.it; d.tellogamarra@sssup.it

the third section explains simulations developed on the application of forward models, the fourth section shows the general architecture for the reaching task; in the fifth section the method for using the forward model for the reaching has been described; finally, conclusions are explained in the sixth section.

2. Theoretical background

2.1. Adaptive neuro-fuzzy inference system (ANFIS)

In order to set the theoretical background that is necessary to better understand this paper, the adaptive neuro-fuzzy inference system (ANFIS) is described. ANFIS is a neural network that we have used to construct the forward model.

Here the ANFIS (Jang 1993) is reviewed in brief. The adaptive neuro-fuzzy inference systems are fuzzy sugeno models put into the framework of adaptive systems, and are composed of the following rules:

Rule 1: if $\times 1$ is A1 and $\times 2$ is B1, then
 $f_1 = a_1 \times 1 + b_1 \times 2 + c_1$.

Rule 2: if $\times 1$ is A2 and $\times 2$ is B2, then
 $f_2 = a_2 \times 1 + b_2 \times 2 + c_2$.

Figure 1 illustrates the architecture of the network. In the first layer, the degree of membership of the input is computed using a Gaussian membership function:

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

where a_i , b_i and c_i are the parameters of the Gaussian function. The second layer calculates the firing strength (or weight) w_i of the i th rule

$$w_i = \mu_i(x_1) \mu_i(x_2). \quad (2)$$

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

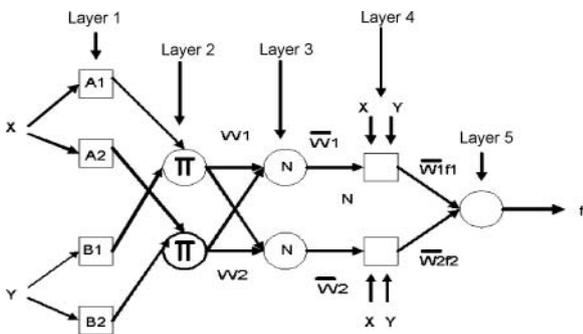


Figure 1. ANFIS architecture.

$$\bar{w} = \frac{w_i}{w_1 + w_2}. \quad (3)$$

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

$$\bar{w}_i f_i = \bar{w}_i (p_i x + q_i y + r_i). \quad (4)$$

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

$$\sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}. \quad (5)$$

Training the network consists of finding suitable parameters for layers 1 and 4. Gradient descent methods are typically used for the non-linear parameters of layer 1, while batch or recursive least squares are used for the linear parameters of layer 4 or even a combination of both (see Jang (1993) for details).

Now that the ANFIS neural network has been described, it is necessary to underline that the ANFIS is used to construct a forward model. The forward model receives as inputs the manipulator joints and it has as output the coordinates of the end-effector in the image plane. The forward model delivers the image Jacobian that is used in the visual servo control.

2.2. Visual servoing

When a robotic manipulator is used for making a reaching task, the traditional approach identifies the target in the 3-D space and draws a spline curve. The curve could be a polynomial of different degrees between the start point and the final point (target). In order to get every point of the trajectory, the robot uses an interpolation.

The other approach for the reaching task is visual servoing. Visual servoing uses vision to identify the target and a control law based on vision delivers every point of the trajectory that must follow the robot to arrive the target.

The servoing is a synergy of different engineering fields such as control, vision and robotics. In Hutchinson et al. (1996), we can find a survey about the methods and techniques used in visual servoing. The controller implemented in this paper is classified, according to Hutchinson et al. (1996), as an Image-Based Visual Servoing technique (IBVS), because the error is minimised in the image plane and not in the three-dimensional space. If the error could be minimised in the three-dimensional space, it would be necessary to reconstruct from the images the position of the end-effector in the three-dimensional workspace, an approach that is known as Position-Based Visual Servoing (PBVS). In addition, our servoing algorithm is classified in the literature as an eye-to-hand scheme, because the camera is not mounted in the end-effector.

The algorithm used for the visual servoing is the one based on Armstrong et al. (1999). Piepmeier (1999) used a dynamic Gauss-Newton method to minimise the errors in the image plane. The error for a static target is defined as the difference between the position in the image plane of the target y^* and the end-effector $y(\theta)$.

$$f(\theta) = y(\theta) - y^*. \quad (6)$$

The dynamic Gauss-Newton method computes the joint angles iteratively. At each iteration k the angular position is computed as

$$\theta_{k+1} = \theta_k - (\hat{J}_k^T \hat{J}_k)^{-1} \hat{J}_k^T \left(f_k + \frac{\partial f_k}{\partial t} h_t \right). \quad (7)$$

In Equation (7), the term h_t is a time increment and is defined as $h_t = t_k - t_{k-1}$; the term $\frac{\partial f_k}{\partial t} h_t$ predicts the change in the error function for the next iteration and \hat{J}_k represents an approximation to the Jacobian in the k instant.

$$\hat{J}_k = \hat{J}_{k-1} + \left(\Delta f - \hat{J}_{k-1} h_\theta - \frac{\partial f_k}{\partial t} h_t \right) \times (\lambda + h_\theta^T P_{k-1} h_\theta)^{-1} h_\theta^T P_{k-1}, \quad (8)$$

$$P_k = \frac{1}{\lambda} (P_{k-1} - P_{k-1} h_\theta (\lambda + h_\theta^T P_{k-1} h_\theta)^{-1} h_\theta^T P_{k-1}), \quad (9)$$

where $0 < \lambda \leq 1$ is the forgetting factor, $h_\theta = \theta_k - \theta_{k-1}$ and $\Delta f = f_k - f_{k-1}$. Equations (8) and (9) define the recursive update of \hat{J}_k .

It is necessary to underline that this visual servoing control law proposed by Piepmeier (1999) can also be extended and applied to a case where the target as well as camera are moving, and that could happen when the robot head moves eventually. The visual servoing strategy used here does not need any calibration procedure; so we avoid the need to calculate the extrinsic and intrinsic parameters of the camera and also do not need any triangulation procedure for a stereo-vision system. These nice properties of the algorithm lead us to directly apply the algorithm to the robot.

3. Simulation of forward models

In order to show how forward models could help the performance of a visual servoing task, simulations were developed in Matlab. It is worthwhile to describe briefly these simulations that served as a prototype of how forward models could be used in a reaching task. This previous work is detailed in Pinpin et al. (2008) and here we summarise the procedure and the results we obtained, as these were the first steps for the development of our approach and served as the framework for this work.

3.1. Experimental setup

The Robotics Toolbox of Peter Corke for Matlab (Corke 1996) was chosen as a simulation platform. The simulated PUMA 560 manipulator is used in this work. For the vision system, the Epipolar Geometry Toolbox for Matlab (Marrionini and Prattichizzo 2005) is used to simulate two fixed cameras.

3.2. Forward model construction

To obtain a forward model the robot went through a motor babbling phase. In this exploratory phase, the angular positions of the robot joints and the end-effector position in the visual system were recorded.

The data collected from the babbling phase is used to create a forward model of the robot. 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 clustered using the unsupervised clustering algorithm of the toolbox that uses the subclustering algorithm (Yager and Filev 1994). The unsupervised clustering algorithm gives the initial structure of the network (number of fuzzy rules) and parameters (initial parameters of the gaussian membership functions).

A total of four ANFIS neural networks have been constructed – one ANFIS for each image features coordinate u_L, v_L, u_R and v_R . Each neural network has nine inputs ($q_0, q_1, q_2, q_3, q_4, q_5, p_x, p_y$ and p_z). The first six inputs are the angular positions of the joints of the manipulator and the other three 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 and p_z) in one of the cameras (left or right).

3.3. Initial image Jacobian estimation

The forward model encoded in the ANFIS networks is used in obtaining an estimate of the initial image Jacobian of the manipulator for a given joint position. To obtain the initial estimation of the Jacobian, a virtual perturbation of the manipulator joints at the current position is done using the ANFIS networks. Each joint is individually perturbed and the resulting changes of the feature points are used to initialise the corresponding column of the image Jacobian. The changes in the image feature points are computed using the forward model instead of the cameras, that is, the joint angles and the coordinates of each of the five tracked points are inserted as inputs to our forward model. The output of the forward model gives the position each of the end-effector points in the image planes of the ‘robot eyes’. The robot virtually perturbs its visuo-motor map (forward model). This map is a kind of mental abstraction of how a

human being encodes a learning process. The robot, thanks to the forward model built with the ANFIS neural networks, has an initial estimate of its own dynamic visuo-motor map (image Jacobian) based on its own history that can help him to reach a desired position.

3.4. Forward models improve visual servoing

The final objective of the robot is that it could reach a target in a specific position of its workspace. The initial estimate of the image Jacobian is used at the beginning of the visual servoing controller.

The manipulator practically starts in a position opposed to the target. The number of iterations for the control loop is equal to 600 iterations. The robot has as initial coordinates in joint space $(-3.0252, 0.07757, -1.5126000)$ radians and the desired position of the target in joint space coordinates is (0.9300000) radians. Figure 2 shows 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 3 shows the position of the robot when it has reached the target.

In order to test the validity of the use of the forward model to estimate an initial image Jacobian a comparison with random image Jacobians is made. The servo-controller

is tested with 10 different random initial image Jacobians. Each element of each random image Jacobian (a 20×6 matrix) is initialised with values from the range (-11) . Figure 4 shows the results obtained with these random image Jacobians (blue curves) and the one estimated using the forward model (red ones). This simulation shows that the error obtained with the initial Jacobian derived from the ANFIS forward models has a better performance and is matched only by two random Jacobians.

4. General architecture for the robotic implementation of visual servoing

Our system architecture was defined based on these previous results. For the iCub implementation, it was not necessary to go through a complete motor babbling phase in the entire workspace as we did with the PUMA 560. Our main concern is to have an initial image Jacobian and we discovered that we could obtain good results from perturbing a forward model trained with very few reaching attempts. Even if these reaching attempts could not be successful, they are useful in constructing the forward model because these reaching attempts contain information that comes from the robot’s proprioceptive system in the form of encoder readings and information derived from the visual system. The

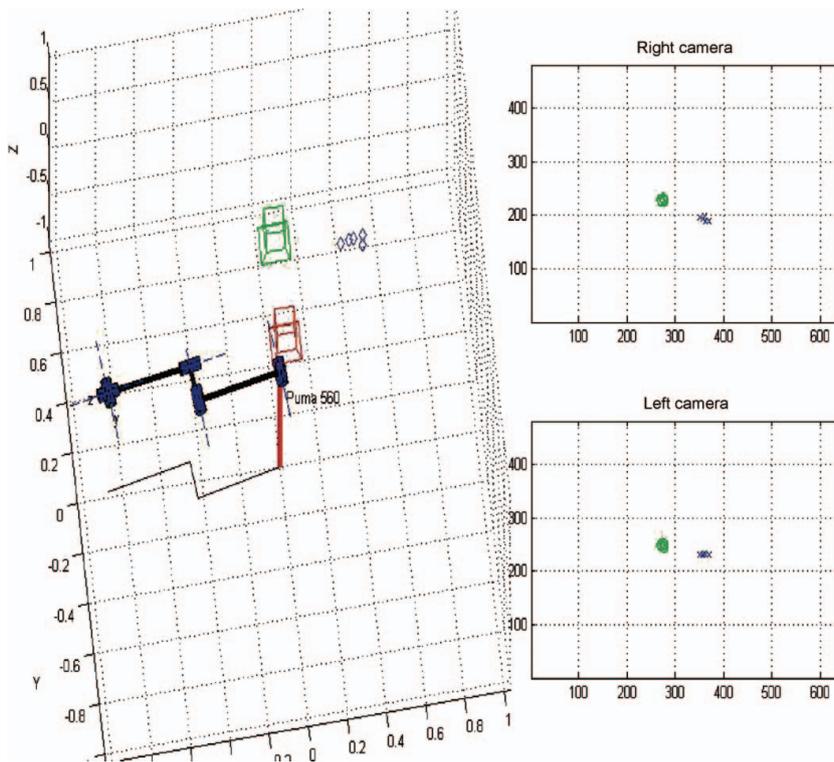


Figure 2. The left subplot has the position of the robot at the beginning of the reaching. The target position is represented by the blue points in the left subplot (feature points of the end-effector at the end of reaching). The right top and right bottom subplots show the end-effector position (green circles) and the target image coordinates (blue crosses).

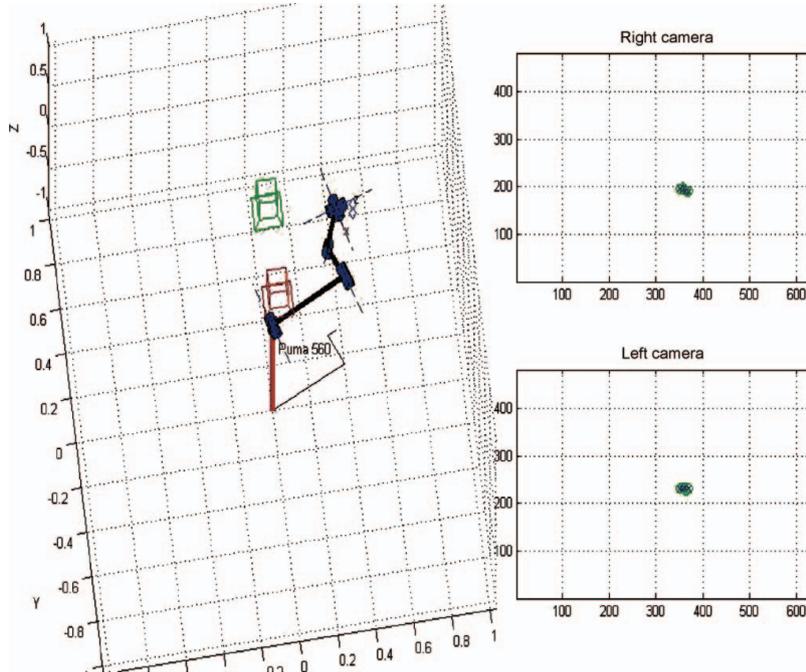


Figure 3. The adaptive sinusoidal left subplot has the position of the robot at the end of the reaching. The target is represented by the blue points in the left subplot (partially obscured by the end-effector). The right top and right bottom subplots show the end-effector position (green circles) and the target (blue crosses). Since the robot end-effector has reached the target, the blue crosses are overlapping with green circles.

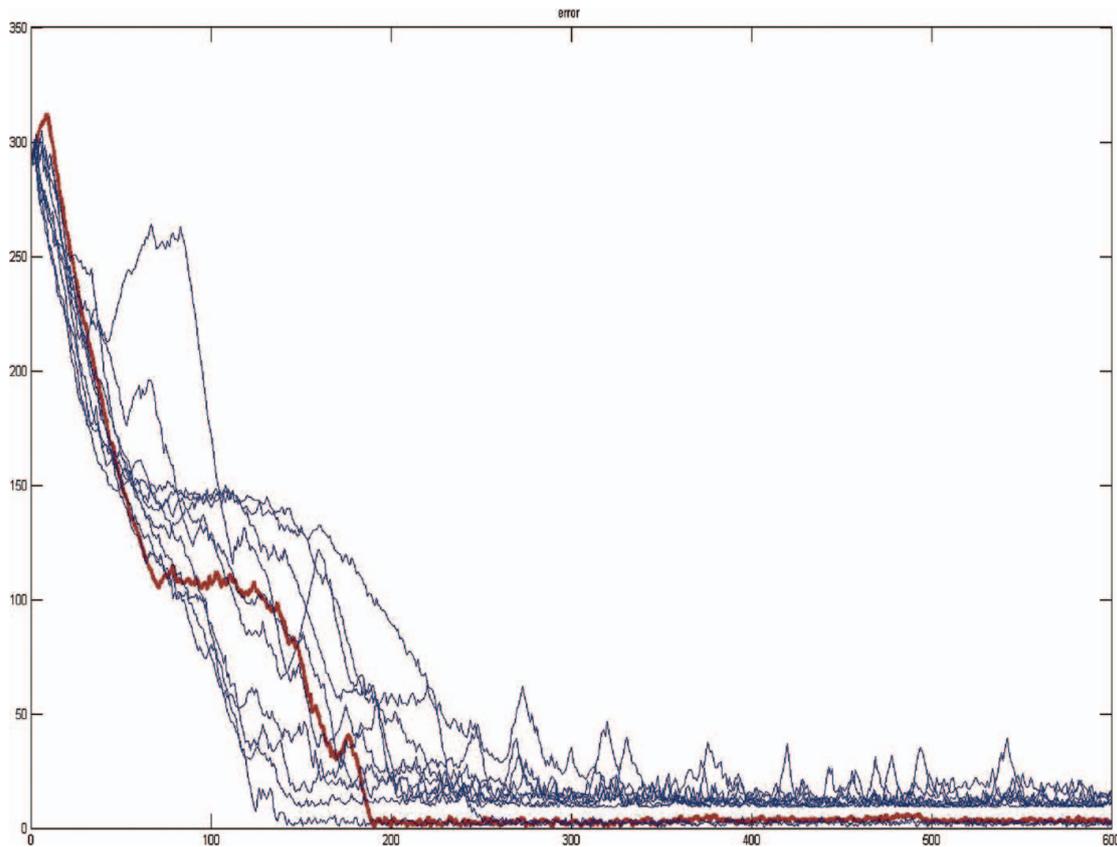


Figure 4. Comparison of servoing error performance using random image Jacobians (blue) and the image Jacobian estimated from the ANFIS forward model (red). Each curve represents the norm of the error vector in pixels at each time step.

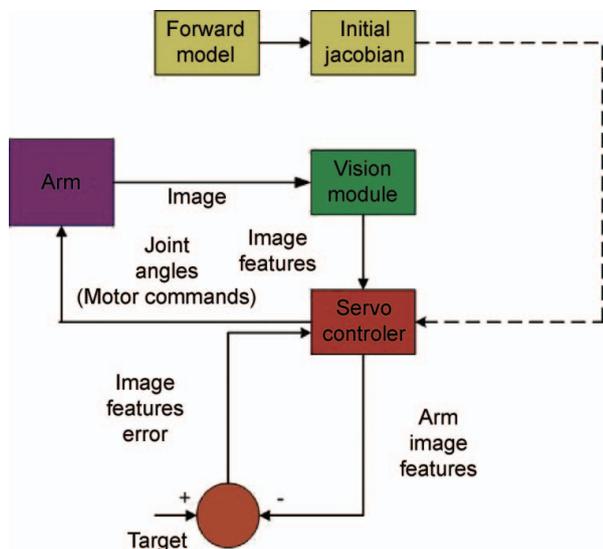


Figure 5. General architecture for a vision-based reaching using a forward model.

forward model is constructed using ANFIS neural networks as in the case of the PUMA 560.

Figure 5 shows the general architecture that is used for vision-based reaching. The forward model is updated and consequently perturbed at the beginning of the visual servoing loop. The next section explains the implementation of this method in the iCub setup.

5. Experiments with the icub simulator

5.1. Experimental platform setup

5.1.1. RobotCub platform

The RobotCub platform is the result of a research project aimed to develop a robotic child (iCub) with the physical (height 90 cm, mass less than 23 kg and 53 d.o.f.) and ultimately cognitive abilities of a 2.5-year-old human child. The iCub is a freely available open system, which can be used by scientists in all cognitive disciplines from developmental psychology to humanoid robotics to enhance understanding of cognitive systems through the study of cognitive development. The iCub is open source and open hardware (mechanical and electronic design).

One of the milestones of the RobotCub philosophy on cognition is the belief that manipulation plays a fundamental role in the development of cognitive capability. The iCub will test this hypothesis acting in cognitive scenarios, performing tasks useful for learning and interacting with the environment and humans. The capacity for cognitive development in the iCub is a fundamental difference from the many excellent humanoids already developed as mentioned in Tsagarakis et al. (2007). Figure 6 shows the robotic platform, the iCub.

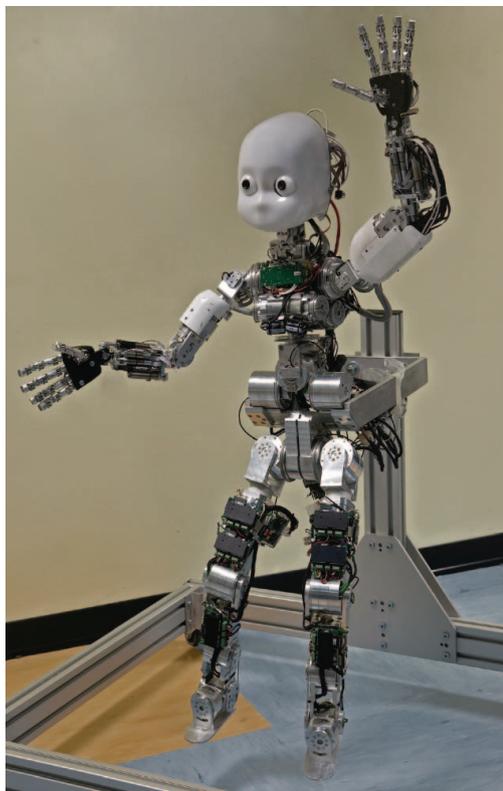


Figure 6. The iCub humanoid robot.

5.1.2. Software architecture

Yet Another Robot Platform (YARP) software described in Metta et al. (2006) and Fitzpatrick et al. (2008) is the middleware software used by the iCub humanoid robot. YARP is an open-source project for a long-term software development for applications that are real-time, computation-intensive and involve interfacing with diverse and changing hardware.

YARP's goal is to minimise the effort devoted to software development by facilitating code reuse and modularity, and so maximise research-level development and collaboration. In short, the main features of YARP include support for interprocess communication and image processing as well as a class hierarchy to ease code reuse across different hardware platforms. In addition, YARP facilitates the implementation of a distributed controller in a cluster. YARP is currently used and tested on Windows, Linux, MacOS and Solaris, which are common operating systems used in robotics.

5.1.3. The iCub simulator

In this stage of our research work, the algorithms were tested in the iCub simulator that was developed by Vaddim Tikhanoff and is shown in Figure 7. The simulator as stated in Tikhanoff et al. (2008a; 2008b) has been designed to

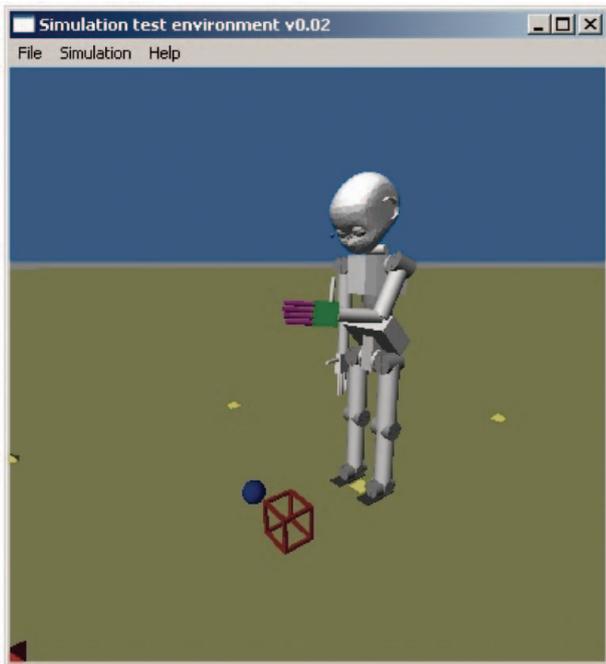


Figure 7. The iCub ODE simulator.

reproduce, as accurately as possible, the physics and the dynamics of the robot and its environment. It has been constructed collecting data directly from the robot design specifications in order to achieve an exact replication of the iCub. This means the same height, mass and d.o.f.

The iCub simulator was created using open source libraries. It uses the open dynamics engine (ODE) for simulating rigid bodies and the collision detection algorithms to compute the physical interaction with objects. ODE consists of a high-performance library for simulating rigid body dynamics using a simple C/C++ API. The iCub simulator also uses YARP as its software architecture. It is worth mentioning that as stated in Tikhanoff et al. (2008a), the iCub simulator is one of the few that attempts to create 3-D dynamic robot environment capable of recreating complex worlds and fully based on non-proprietary open source libraries.

The experiments done in this paper were tested in the iCub simulator. Otherwise, the babbling phase would require a collision detection to prevent self-collision in the robot, which still has not been incorporated in the robot; this issue is absent from a standard robot manipulator, such as the PUMA 560, which does not have a body to collide with the end-effector.

5.1.4. The vision system

In order to implement visual servoing we needed to develop a vision system that could take the images from both simulated cameras and segment the end-effector in the image

plane. A segmentation based on color appeared as a natural and simple choice; it is necessary to underline that the precision and the performance of the servoing is dependent on the quality of features provided to the controller.

The vision system module detects image features of the end-effector (colored hand) in the image field of view. It uses an algorithm based on blob color detection. The hand color of the iCub that was originally gray was changed to green and violet, as can be seen in Figures 7 and 8 (figures available in colour online). The visual blobs module that is a part of the iCub open source software was developed by Jonas Ruesch in order to construct a salience map for the iCub attention system as referred to in Ruesch et al. (2008). This module was modified in order to segment and filter based on color. Figure 8 shows the image of the robot's hand captured from the left camera, before applying the color segmentation and blob detection.

Figures 9 and 10 show the blobs obtained from the segmented images of the left hand in both the cameras. The images are flipped because that is the native memory format in which the simulator delivers the images from the simulated cameras and in the next step they are rotated in 180° . In order to gain some computational time the images were not flipped right side up. The image features generated for the vision algorithm are the centroids and the area of the blobs.

5.1.5. Forward model creation

The data collected from a single-reaching attempt is used to create a forward model of the robot. This attempt does not

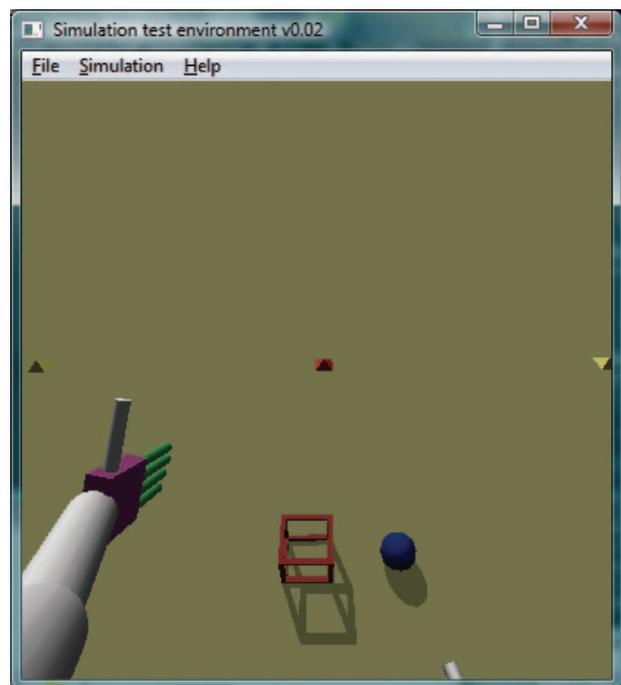


Figure 8. Left hand of the iCub taken from the left camera.

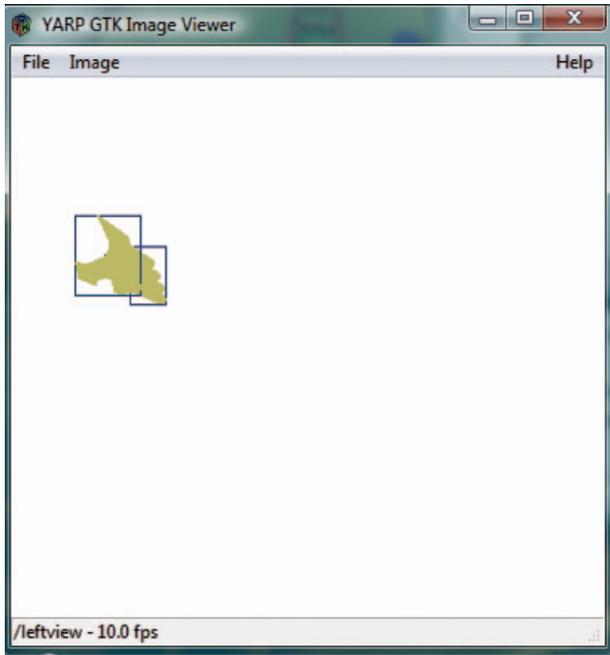


Figure 9. View of the left camera image segmented with two detected blobs.

matter if the reaching was successful or not. It just serves to create an approximation of the model of the system. ANFIS neural networks are used for constructing the forward model.

The training data is a set that includes the joint angles of the manipulator and the end-effector features in the im-

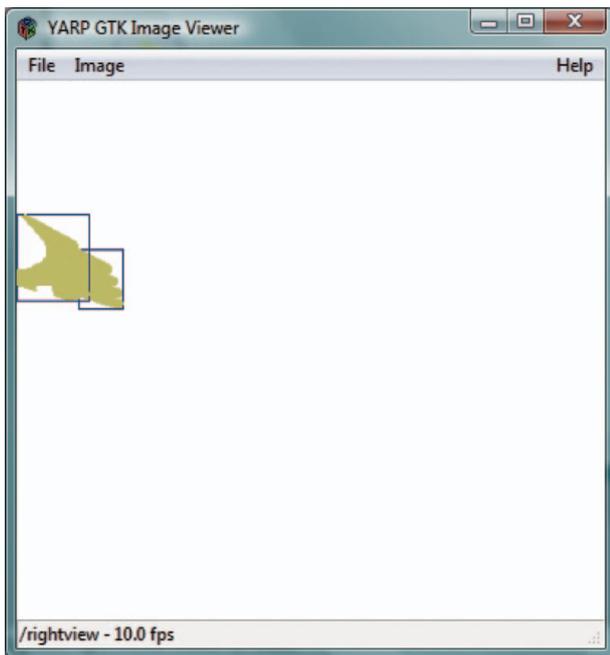


Figure 10. View of the right camera image segmented with two detected blobs.

ages, the features that were used were the blob centroid in image coordinates, and the blob's area. Because we have two blobs (green and violet), we had six image features for each camera. The input data is clustered using the unsupervised clustering algorithm of the toolbox that uses the subclustering algorithm (Yager and Filev 1994).

A total of 12 ANFIS neural networks have been constructed – one ANFIS for each image feature. Each neural network has seven inputs ($q_0, q_1, q_2, q_3, q_4, q_5$ and q_6). These are the angular positions of the joints of the manipulator. The output of the network is an image feature of the end-effector in one of the cameras (left or right). A total of 12-image feature has been tracked because it has been seen in the simulations that as the number of image features is increased the robustness of the algorithm grows.

The number of rules created as an average for the ANFIS neural network was 11 with a subclustering radius of 0.3. This subclustering parameter tunes the number of fuzzy rules constructed. There is a tradeoff for choosing the value of this parameter because if there are more rules, the computational burden increases. The training was done using a hybrid method that is a combination of back-propagation and recursive least square algorithms. The 12 ANFIS neural networks were trained just for 20 epochs.

5.2. Reaching task results

5.2.1. Initial image jacobian estimation

The estimated initial image Jacobian of the manipulator for a given joint position is obtained from the ANFIS neural network that represents the forward model. First, the ANFIS networks are updated with the value of the current joint position and the feature vector obtained from the vision system. This results in an improved forward model at the current robot position. Then, to obtain the initial estimation of the image Jacobian, a virtual perturbation of the manipulator joints at the current position is done using the ANFIS networks, as we have seen previously in the Matlab simulations using the PUMA robot. The changes in the end-effector image features are computed using the forward model instead of the cameras. The joint angles are inserted as inputs to our forward model. The output of the forward model gives the desired image feature of the blob generated by the end-effector (centroids and blob areas) in the image planes of the cameras.

5.2.2. Improved visual servoing performance

The initial estimate of the image Jacobian is used at the beginning of the visual servoing controller. The manipulator starts practically in a position opposed to the target as was done with the PUMA. The variation of the joint velocities of the manipulator has been clamped between $-60^\circ/\text{sec}$ and $60^\circ/\text{sec}$. The camera frame rate for this simulation is fixed

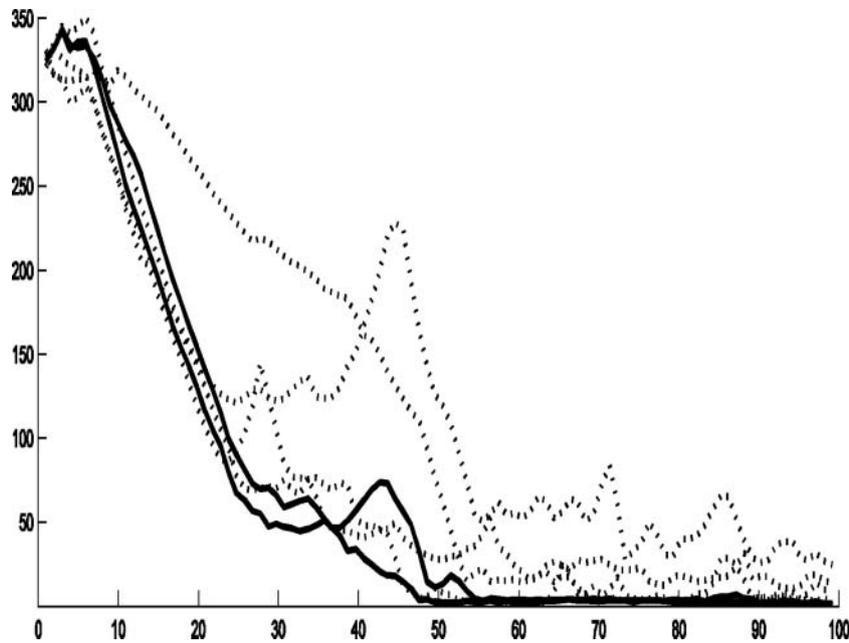


Figure 11. Comparison of servoing error performance using random image Jacobians (dashed) and the image Jacobian estimated from the ANFIS forward model (solid). Each curve represents the norm of the error vector in pixels at each time step.

at 1/60 sec. The simulation time step is 0.005. The number of iterations for the control loop is equal to 100 iterations.

For the sample-reaching trials as shown in Figure 11, the robot has as initial joint coordinates of $-25^\circ, 30^\circ, -10^\circ, 20^\circ, 0^\circ$ and 0° and the desired position of the target in joint space coordinates is $65^\circ, 10^\circ, 0^\circ, 20^\circ, 0^\circ$ and 10° .

A comparison between the initial image Jacobians obtained from the forward model and random image Jacobians is made. The servo-controller is tested with five different random initial image Jacobians. Each element of random image Jacobian (a 12×7 matrix) is initialised with the values from the range $[-11]$. Figure 11 shows the results obtained with these random image Jacobians (dashed curves) and the one estimated using the forward model (solid curves). This simulation shows that the errors obtained with the image Jacobians obtained from the forward model are matched in performance only by one random image Jacobian. We also note that some random image Jacobians became singular so that the servoing loop did not finish.

6. Conclusion

This paper describes how forward models can be used in a reaching task using visual servoing. The method consists in constructing the forward model from a first reaching attempt and subsequently updated before succeeding attempts. The forward model is constructed using ANFIS neural networks. Perturbing the forward model to obtain an initial image Jacobian improves an image-based visual controller.

It is interesting to note that compared to Sun and Scasellati (2005), in which a robot Jacobian is constructed using radial-basis function neural networks and trained offline, our approach opens the possibility to train online the forward model and enrich it with every reaching attempt. Also, for the visual servoing we are not using resolved motion rate control that depends on a static inverse Jacobian, which in turn requires accurate kinematic models; instead, the method we used for the servoing requires an inverse image Jacobian that is updated with each frame using the Broyden update method. Thus, our approach takes advantage of the method's tolerance to noise, does not require calibration of robot kinematics or camera parameters and improves on it by giving it a better starting image Jacobian.

We have demonstrated the potential use of a forward model similar to how humans take advantage of a sensory motor map constructed through previous reaching attempts. The approach that is shown here enables the robot with a virtual sensory-motor map encoded in a neural network, which is perturbed to get an initial estimation of its dynamic visuo-motor relationship (image Jacobian) to start the reaching movement.

Finally, we consider our implementation as one of the few works in which visual servoing is applied in a humanoid robotic platform with a stereo-vision system.

Acknowledgment

This work is supported by the ROBOTCUB project (IST-2004-004370), funded by the European Commission through Unit E5

'Cognitive SYSTEMS'. Also, we would like to thank the Italian Institute of Technology for its financial support to the Ph.D. courses of the first two authors.

References

- Corke PI. 1996. A robotics toolbox for MATLAB. *IEEE Rob Autom Mag.* 3(1):24–32.
- Dearden A, Demiris Y. 2005. Learning forward models for robots. In: 2005 Proceedings of International Joint Conferences of AI, IJCAI: Edinburgh. International Conference on Artificial Intelligence (IJCAI) '05. 30 July to 5 August 2005. Edinburgh, Scotland.
- Fitzpatrick P, Metta G, Natale L. 2008. Towards lng-lived robot genes. *IEEE Rob Autom Mag.* 56(1):29–45.
- Hutchinson S, Hager GD, Corke PI. 1996. A tutorial on visual servo control. *IEEE Trans Rob Autom.* 12(5):651–670.
- Jang JSR. 1993. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Trans Syst, Man and Cybern.* 23(3):665–685.
- Kawato M. 1999. Internal models for motor control and trajectory planning. *Curr Opin Neurobiol.* 9(6):718–727.
- Mariottini GL, Prattichizzo D. 2005. EGT for multiple view geometry and visual servoing: robotics vision with pinhole and panoramic cameras. *IEEE Rob Autom Mag.* 12(4):26–39.
- Metta G, Fitzpatrick P, Natale L. 2006. YARP: yet another robot platform. *Int J Adv Rob.* 3(1):43–48.
- Piepmeyer JA, McMurray GV, Lipkin H. A dynamic Jacobian estimation method for uncalibrated visual servoing. In: GV McMurray editor. *Proceedings of the IEEE International Conference on Advanced Intelligent Mechatronics (ASME)*. 19–23 September 1999. Atlanta, GA, USA.
- Pinpin LK, Tello Gamarra DF, Laschi C, Dario P. 2008. Forward model creation in a six-link manipulator. In: *Biomedical Robotics and Biomechatronics IEEE (BioRob)*, Scottsdale-USA. 19–22 October 2008. Scottsdale, AZ, USA.
- Rao RPN, Shon AP, Meltzoff AN. 2004. Bayesian model of imitation in infants and robots. In: *Imitation and social learning in robots, humans, and animals: behavioural, social and communicative dimensions*. Cambridge: Cambridge University Press.
- Ruesch J, Lopes M, Bernardino A, Hornstein J, Santos-Victor J, Pfeifer R. 2008. Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub. In: 2008 IEEE International Conference on Robotics and Automation (ICRA). 19–23 May 2008. Pasadena, CA, USA.
- Sturn J, Plagemann C, Burgard W. 2008. Unsupervised body scheme learning through self-perception. In *Proceedings of the IEEE International Conference on Robotics and Automation 2008*, Pasadena, California.
- Sun G, Scasellati B. 2005. A fast and efficient model for learning to reach. *Int J Hum Rob.* 2(4):24–32.
- Tikhonoff VFP, Metta G, Natale Lorenzo, Nori F, Cangelosi A. 2008a. An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator. In: *Workshop on Performance Metrics for Intelligent Systems*, National Institute of Standards and Technology Washington DC, USA. 19–21 August 2008.
- Tikhonoff VFP, Nori F, Natale L, Metta G, Cangelosi A. 2008b. The iCub humanoid robot simulator. In: *IROS Workshop on Robot Simulators Nice*, France. 22 September 2008. Nice, France.
- Tsagarakis NG, Metta G., Sandini G., Vernon D, Beira R, Becchi F, Righetti L, Santos-Victor J, Ijspeert AJ, Carrozza MC, Caldwell CG. 2007. Icube the design and realization of an open humanoid platform for cognition and neuroscience research. *J Adv Rob.* 2(10):1–25.
- Wolpert DM, Miall RC, Kawato M. 1998. Internal models in the cerebellum. *Trends Cogn Sci.* 2(9):338–347.
- Yager RR, DP Filev. 1994. Generation of fuzzy rules by mountain clustering. *J Intell Fuzzy Syst.* 2(3):209–219.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

