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

Abstract: 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 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. Although this paper is one of the few attempts of 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 in (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 feedforward motor commands from 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). Roboticists, 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 derived analytically 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. Also in (Sturn J. et al. 2008) a forward model of a robot manipulator is learned using a Bayesian network, after a babbling phase using a monocular camera.

Following a developmental robotics roadmap, this work tries to shed some light in 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 that determines the maximum performance attainable in a reaching 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 and 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 offline training is avoided. ANFIS neural networks 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.

Corresponding author:
Daniel Fernando Tello Gamarra:
e-mail:fgamarra@sssup.it

The remainder of the paper is as follows. In the second section are described the algorithms used in this work; 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 is described; finally, conclusions are explained in the sixth section.

1. Theoretical background

1.1. Adaptive Neuro-Fuzzy Inference System (ANFIS)

In order to set the theoretical background that is necessary to understand better the paper. The ANFIS is described. ANFIS is the neural network that we have used to construct the forward model.

The ANFIS (Jang 1993) is reviewed here briefly. Adaptive Neuro-Fuzzy Inference Systems are Fuzzy Sugeno models put in the framework of adaptive systems, and are 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

Figure 1 illustrates the architecture of the network. In the first layer the degree of the 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 normalized with the sum of all rule's firing strengths:

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

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

$$\overline{w}_i f_i = \overline{w}_i\left(p_i x + q_i y + r_i\right) \quad (4)$$

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

$$\sum_i \overline{w} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (5)$$

Training the network consists of finding suitable parameters for layer 1 and layer 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.

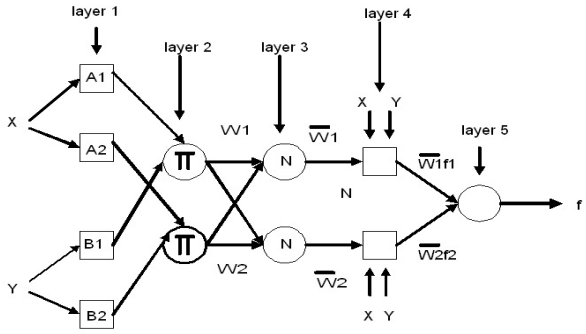

Figure 1 ANFIS architecture.

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.

1.2. 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 of the robot is used 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 to 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, Hager and Corke 1996) as an Image Based Visual Servoing technique (IBVS) because the error is minimized in the image plane and not in the three dimensional space. If the error would be minimized in the three dimensional space would be necessary to reconstruct from the images the position of the end effector in the three dimensional workspace, an approach called Position Based Visual Servoing (PBVS). Also 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 Piepmeier et al. 1999). Piepmeier used a dynamic Gauss-Newton method to minimize the errors in the image plane. The error for a static target is defined as the difference of 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

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

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 and $\hat{J}_k$ represents an approximation to the Jacobian in the k instant.

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

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

Where $0 < \lambda \leq 1$ is the forgetting factor, $h_\theta = \theta_k - \theta_{k-1}$, $\Delta f = f_k - f_{k-1}$. 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 can also be extended and applied to a case in which the target is moving and the camera is moving that would happen if the robot head would eventually move. 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 we do not need any triangulation procedure for a stereo-vision system. These nice properties of the algorithm lead us to apply directly the algorithm to the robot.

2. 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 summarize the procedure and the results we obtained because they were the first step on the development of our approach and served as the framework of this work.

2.1. 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 (Mariottini and Prattichizzo 2005) is used to simulate two fixed cameras.

2.2. 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 R. and D. 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 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).

2.3. 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 initialize 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 human beings encode 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.

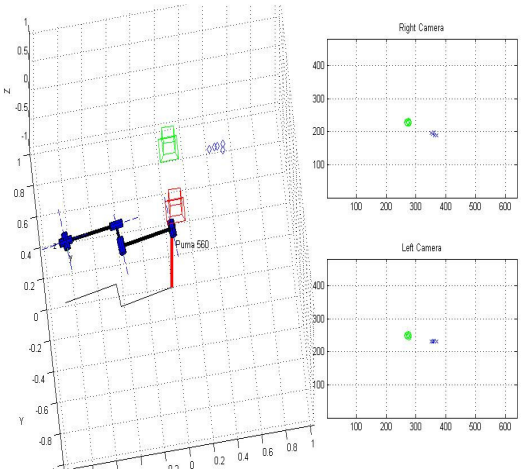

Figure 2 The left subplot has the position of the robot at the beginning of the reaching. The target position is represented by the blue points in the left subplot (feature points of the end effector at the end of reaching). The right top and right bottom subplots show the end-effector position (green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

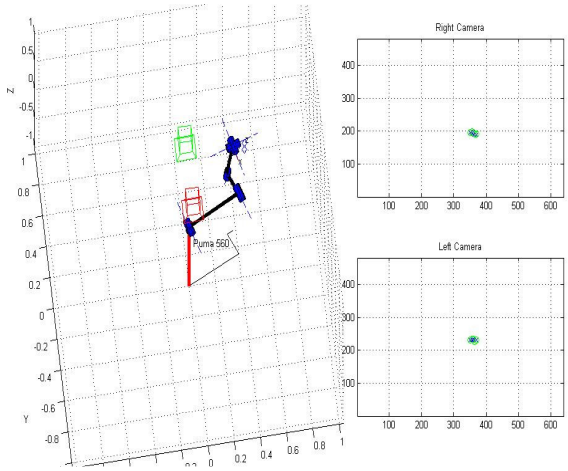

Figure 3 The Adaptive sinusoidal left subplot has the position of the robot at the end of the reaching. The target is represented by the blue points in the left subplot (partially obscured by the end effector). The right top and right bottom subplots show the end-effector position (green circles) and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

2.4. 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 starts practically 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 -1.5126 0 0 0] radians and the desired position of the target in joint space coordinates is [0.93 0 0 0 0 0] 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 random image Jacobian (a 20 x 6 matrix) is initialized with values from the range [-1 1]. Figure 4 shows the results obtained with these random image Jacobians (blue curves) and the one estimated using the forward model (red one). 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.

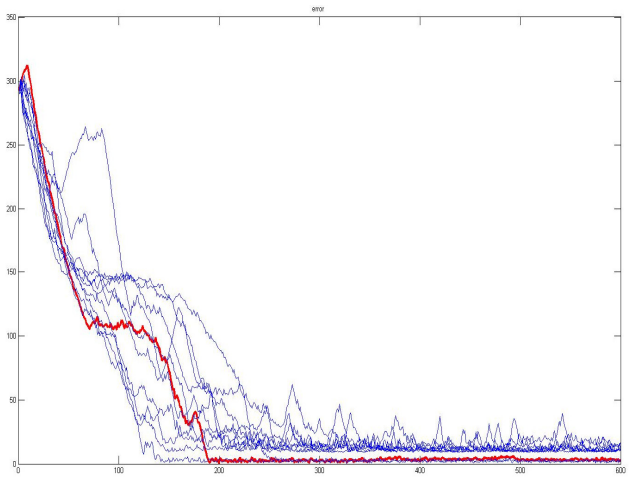

Figure 4 Comparison of servoing error performance using random image Jacobians (blue) and the image Jacobian estimated from the ANFIS forward-model (red). Each curve represents the norm of the error vector in pixels at each time step.

3. 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 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.

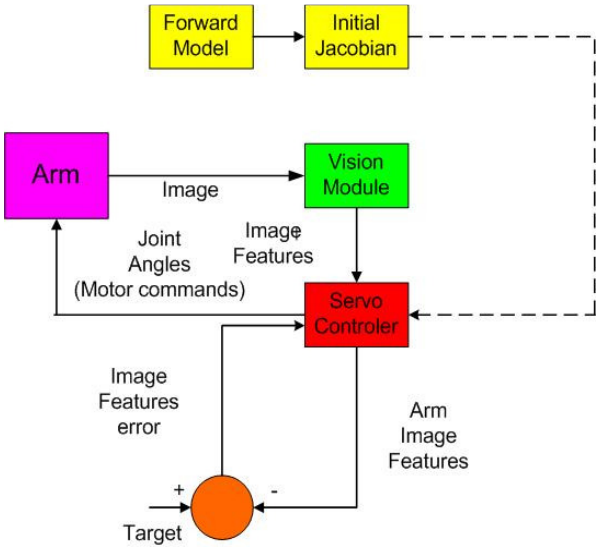

Figure 5 General Architecture for a vision-based reaching using a forward model.

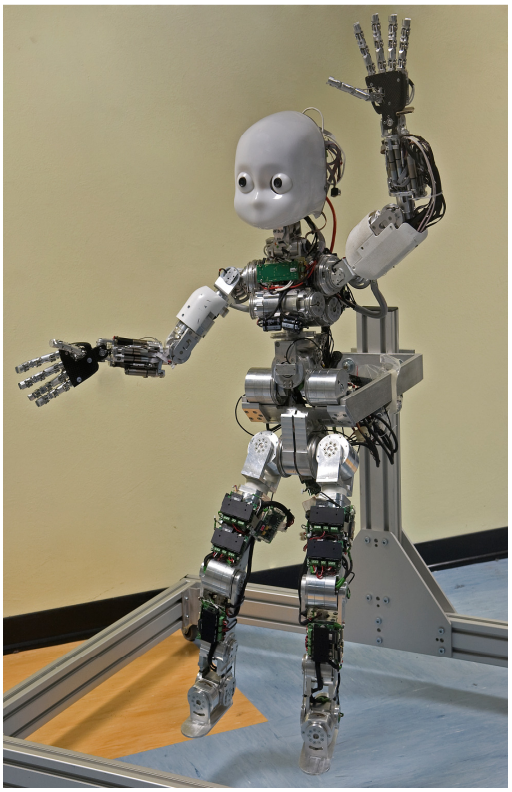

Figure 6 ICub humanoid robot.

4. Experiments with the icub simulator

4.1. Experimental platform setup

4.1.1. 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 of d.o.f.) and ultimately cognitive abilities of a 2.5 years-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 N. G. 2007). Figure 6 shows the robotic platform, the iCub.

4.1.2. Software architecture

YARP (Yet Another Robot Platform) software described in (G. Metta 2006) and (Fitzpatrick 2008) is the middleware software used by the iCub humanoid robot. YARP is an open-source project for 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 minimize the effort devoted to software development by facilitating code reuse and modularity, and so maximize 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. Also 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.

4.1.3. 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 V. 2008a, Tikhanoff V. 2008b) has been designed to 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 same height, mass and d.o.f.

The iCub simulator was created using open source libraries. It uses ODE (Open Dynamics Engine) 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 V. 2008a), the iCub simulator is one of the few that attempts to create 3D 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

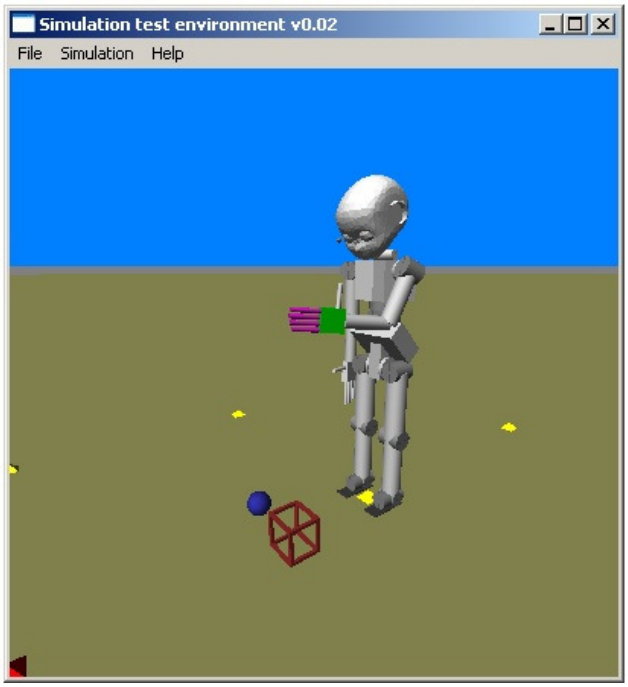

Figure 7 ICub ODE simulator.

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

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.

4.1.4. 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 the features provided to the controller.

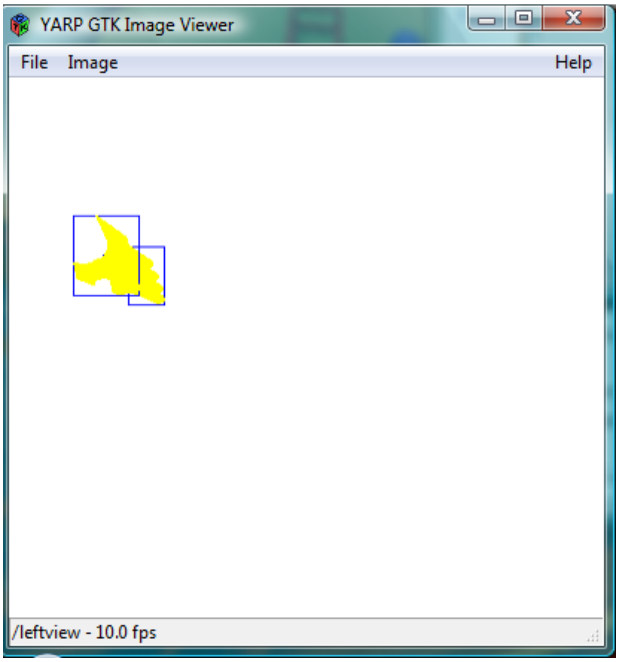

Figure 9 View of the left camera image segmented with two detected blobs.

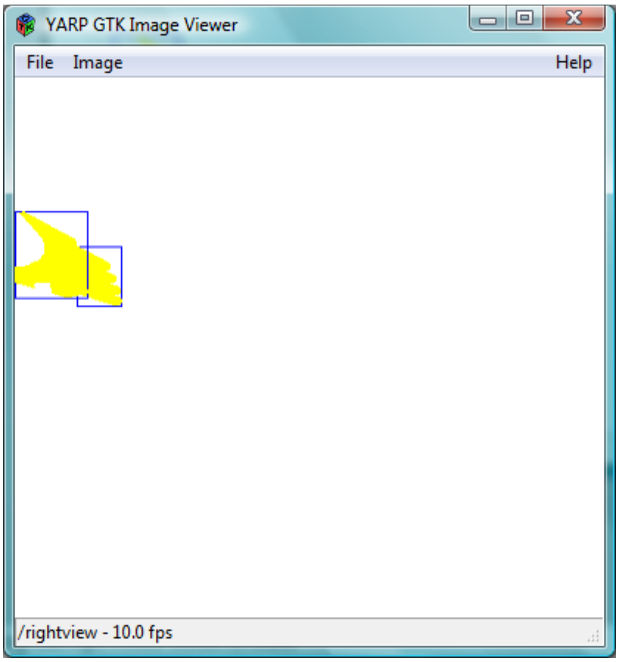

Figure 10 View of the right camera image segmented with two detected blobs.

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. The visualblobs module that is 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 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 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 degrees. 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.

4.1.5. 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 matter if the reaching was sucessful 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 images, the features that were used were the blob centroid in image coordinates, and the blob's area. Since we have two blobs (green and violet), we had 6 image features for each camera. The input data is clustered using the unsupervised clustering algorithm of the toolbox that uses the subclustering algorithm (Yager R. and D. 1994).

A total of twelve ANFIS neural networks have been constructed – one ANFIS for each image feature. Each neural network has 7 inputs (q_0, q_1, q_2, q_3, q_4, q_5, 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 twelve image feature have 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.

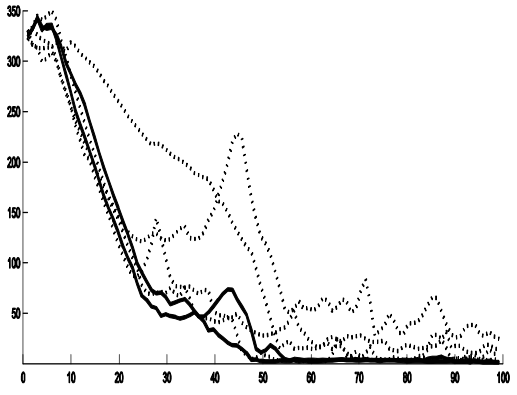

Figure 11 Comparison of servoing error performance using random image Jacobians (dashed) and the image Jacobian estimated from the ANFIS forward-model (solid). Each curve represents the norm of the error vector in pixels at each time step.

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 chosing 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.

4.2. Reaching task results

4.2.1. 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.

4.2.2. 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 and 60 degrees/sec. The camera frame rate for this simulation is fixed 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 shown in Figure 11 the robot has as initial joint coordinates [-25, 30, -10, 20, 0, 0] degrees and the desired position of the target in joint space coordinates is [-65, 10, 0, 20, 0, 0, 10] degrees .

A comparison between the initial image Jacobians obtained from the forward model and random image Jacobians is made. The servo-controller is tested with 5 different random initial image Jacobians. Each element of random image Jacobian (a 12 x 7 matrix) is initialize with values from the range [-1 1]. 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 error 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 the servoing loop did not finish.

5. 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 and 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 thorough 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 IIT (Italian Institute of Technology) for its financial support to the Phd courses of the first two authors.

References

Wolpert DM, Miall RC and Kawato M. 1998. Internal models in the cerebellum. Trends in Cognitive Sciences. 2(9):338-347.

Kawato M. 1999. Internal models for motor control and trajectory planning. Curr Opin Neurobiol. 9(6):718-727.

Rao RPN, Shon AP and 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.

Sun G and Scasellati B. 2005. A fast and efficient model for learning to reach. International Journal of Humanoid Robotics. 2(4):24-32.

Dearden A and Demiris Y. Learning Forward Models for Robots. In. 2005 Proceedings of (International Joint Conferences of AI) IJCAI; Edinburgh.

Sturn J., Plagemann C. and W.. B. 2008. Unsupervised Body Scheme Learning through Self-Perception. In: ICRA.

Jang JSR. 1993. ANFIS: adaptive-network-based fuzzy inference system. Systems, Man and Cybernetics, IEEE Transactions on. 23(3):665-685.

Hutchinson S, Hager GD and Corke PI. 1996. A tutorial on visual servo control. Robotics and Automation, IEEE Transactions on. 12(5):651-670.

Armstrong Piepmeier J, McMurray GV and Lipkin H. A dynamic Jacobian estimation method for uncalibrated visual servoing. In: GV McMurray editor. In Proceedings of the IEEE International Conference on Advanced Intelligent Mechatronics (ASME).

Pinpin LK, Tello Gamarra DF, Laschi C and Dario P. 2008. Forward Model Creation in a Six Link Manipulator. In: Biomedical Robotics and Biomechatronics IEEE (BioRob) Scottsdale-USA.

Corke PI. 1996. A robotics toolbox for MATLAB. Robotics & Automation Magazine, IEEE. 3(1):24-32. Available

Mariottini GL and Prattichizzo D. 2005. EGT for multiple view geometry and visual servoing: robotics vision with pinhole and panoramic cameras. Robotics & Automation Magazine, IEEE. 12(4):26-39.

Yager R. and D. F. 1994. Generation of fuzzy rules by mountain clustering. Journal of Intelligent and Fuzzy Systems. 2(3):209-219.

Tsagarakis N. G. SG, Vernon D., Beira R., Becchi F., Righetti L., Santos-Victor J., Ijspeert A.J., M.C. Carrozza and Caldwell C.G. 2007. Icub the design and realization of an open humanoid platform for cognition and neuroscience research. Advanced Robotics:1-25.

G. Metta PF, L. Natale. 2006. YARP:yet another robot platform. International Journal on Advanced Robotics Systems(Special Issue on Software Development and Integration in Robotics). Available

Fitzpatrick P, Metta, G., Natale, L. 2008. Towards Long-Lived Robot Genes. Robotics & Automation Magazine, IEEE. 56(1):29-45.

Tikhanoff V. FP, 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.

Tikhanoff V. FP, Nori F, Natale L. Metta G. , Cangelosi A. 2008b. The iCub Humanoid Robot Simulator. In: IROS Workshop on Robot Simulators Nice, France.

Ruesch J, Lopes M, Bernardino A, Hornstein J, Santos-Victor J and Pfeifer R. Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub. In. 2008 IEEE International Conference on Robotics and Automation (ICRA).