# Learning the velocity kinematics of iCub for model-based control: XCSF versus LWPR

Guillaume Sicard, Camille Salaün, Serena Ivaldi, Vincent Padois, Olivier Sigaud

HAL Id: hal-00624056

https://hal.sorbonne-universite.fr/hal-00624056v2

Submitted on 5 Oct 2011

# Learning the velocity kinematics of ICUB for model-based control: XCSF versus LWPR

Guillaume Sicard, Camille Salaün, Serena Ivaldi, Vincent Padois and Olivier Sigaud

*Abstract*—The model-based control of humanoid robots requires the availability of accurate mechanical models that can be hard to obtain in practice. One approach to this problem consists in calling upon machine learning methods. In this paper, using a standard control approach based on visual servoing, we compare the accuracy of two supervised learning methods, namely LWPR and XCSF, to extract the forward velocity kinematics of the upper body of the ICUB robot. Experiments are performed in simulation, using one arm and the head for reaching tasks. We show that both methods provide accurate models of the robot, with a slight advantage to XCSF over LWPR.

## I. INTRODUCTION

Given the complexity of humanoid robots and of their associated missions, using machine learning techniques for the control of these platforms is becoming mandatory (see [1] for an overview). Among different approaches, trying to learn models of the robot at the kinematic or the dynamic level has recently received a lot of interest (e.g. [2], [3], [4]). The idea consists in extracting these models with general function approximation methods based on input/output data that are recorded while using the robot.

In [5], a control framework combining the Resolved Motion Rate Control (RMRC) framework and learning the Forward Velocity Kinematics (FVK) of the system with LWPR [2] is presented. The specificity of this framework is that it can combine several tasks ranked by priority, either compatible or not. However, this study is limited in several respects.

First, it is based on a simple planar arm model. Here, we evaluate the potential of this approach with the ICUB humanoid robot [6]. In that respect, our robotic set-up and goal share a lot of similarities with the framework presented in [7] that uses JAMES, an ancestor of ICUB, and RFWR, an ancestor of LWPR. Second, in [5] an important motor babbling stage is required to initialize the learned FVK, which would be hardly feasible on a real robot. Here, the motor babbling stage is not necessary anymore. Finally, the focus of the paper is on an empirical comparison between LWPR and XCSF [8], another state-of-the-art function approximation algorithm.

In order to elaborate this comparison, we first study the performance of both learning techniques on 8 sequential

reaching tasks with the simulator of the ICUB robot [9], using a FVK based on the CAD parameters of the robot as baseline.

The paper is organized as follows. In Section II, we present the task, set-up and methods used to perform our comparison. In Section III, we describe the series of evaluations performed and present the results. We discuss these results in Section IV before concluding.

## II. METHODS

In this section, we first describe the robotic set-up. Then, we give a formal presentation of the visual servoing and inverse velocity kinematics problem that we address. Finally, we briefly describe LWPR and XCSF, the learning algorithms used to extract the model out of experimental data.

### A. Robotics set-up and objectives

Our goal is to compare the accuracy of two supervised learning algorithms, LWPR and XCSF, when approximating the FVK of ICUB, a 53 degrees of freedom and 104cm high humanoid robot. ICUB is actuated with DC brushless motors. It was designed in the context of the ROBOTCUB project[1] and is present in several research laboratories in Europe. The ICUB software architecture is based on YARP [10] which provides a powerful abstraction layer allowing, for example, to transparently test most of the code in simulation before actually executing it on the real robot without any additional implementation effort. This is possible because the ICUB rigid body dynamics simulator also uses YARP as its software interface.

In the following experiments, we use an arm and the head of ICUB in different contexts during a reaching task. The shoulder is a 3 degrees of freedom joint with some coupling between degrees of freedom. The decoupling is done with a low level controller. The elbow is a one degree of freedom joint. The neck is composed of two consecutive joints, namely the head yaw and pitch angles. It is actuated with two direct drive DC motors whereas for the arm, transmissions based on cables and pulleys are used.

In total, as illustrated in Fig. 1, we control 4 degrees of freedom for the arm and 2 degrees of freedom for the head[2] while the torso of the robot is attached to a fixed base.

[1]http://www.robotcub.org

[2]see http://eris.liralab.it/wiki/ICub_joints for more information about iCub joints
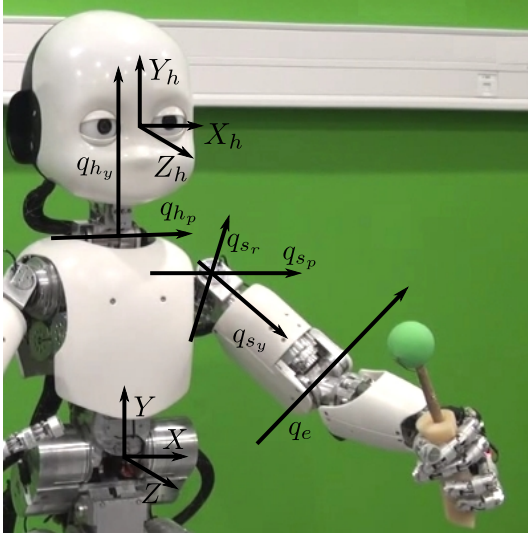
Fig. 1: Parametrization of the arm and head of ICUB. Head, shoulder and elbow controlled axes are represented as well as the head and torso frames.

To learn the FVK in simulation, we simulate the vision process as it would be implemented on the real robot. Furthermore, Gaussian noise is added to the control layer to emulate a physical setup. In order to provide a baseline for the comparison, an analytical expression of this model is obtained using the Kinematic Dynamics Library (KDL) from the Orocos Project [11] using the parameters of the CAD model, which are available in the ICUB wiki[3].

### B. Visual servoing problem formulation

In terms of control, the goal of this paper is to derive, at the velocity kinematics level, a closed-loop controller for the end-effector of the considered kinematic chain (ICUB hand here) based on visual feedback from the cameras of the robot. Our main focus being not vision, we define the end effector of the robot as a green ball that the robot holds with a stick (cf. Fig. 1). We use the Cartesian position of the center of the ball expressed in the head frame as features $\boldsymbol{\xi} = (\xi_x, \xi_y, \xi_z)^T$. This position is reconstructed based on a reprojection software module that uses a ball tracker algorithm based on the OpenCV software [12] as well as on the camera calibration matrices.

The visual servoing problem consists in ensuring the reaching or the tracking of the desired values of the features $\boldsymbol{\xi}^\star = \left(\xi_x^\star, \xi_y^\star, \xi_z^\star\right)^T$. In that case, both the target and the end effector need to be visible at any time. An exponential decrease of the error is achieved by a proportional controller $\dot{\boldsymbol{\xi}}^\star = \lambda\left(\boldsymbol{\xi}^\star - \boldsymbol{\xi}\right)$ where $\lambda$ is a positive definite matrix. In order to design a velocity-level controller for our system, we thus need to relate $\dot{\boldsymbol{\xi}}$ to the articular velocity of the robot.

[3]http://eris.liralab.it/wiki/ICubForwardKinematics

### C. Inverse Velocity kinematics

From the structure of the robot, we know that $\boldsymbol{\xi}$ is a function of both the head configuration $\boldsymbol{q_h} = \left(q_{h_p}, q_{h_y}\right)^T$ and the arm configuration $\boldsymbol{q_a} = \left(q_{s_p}, q_{s_r}, q_{s_y}, q_e\right)^T$, i.e. we have a surjective non-linear function $\boldsymbol{f_\xi}$, namely the forward kinematics, that relates the configuration of the robot $\boldsymbol{q} = \left(\boldsymbol{q_h}^T, \boldsymbol{q_a}^T\right)^T$ to features $\boldsymbol{\xi} = \boldsymbol{f_\xi}\left(\boldsymbol{q}\right)$.

The considered kinematic chain being redundant with respect to the end-effector task, there is an infinite number of possible inverses for $\boldsymbol{f_\xi}$. However, there is no simple method to span the set of possible solutions at the geometric level and this type of mapping is often described at the velocity level. A a matter of fact, we can derive this expression and obtain

$$\dot{\boldsymbol{\xi}} = \underbrace{\left[ \begin{array}{cc} \frac{\partial \boldsymbol{f_\xi}(\boldsymbol{q})}{\partial \boldsymbol{q_h}} & \frac{\partial \boldsymbol{f_\xi}(\boldsymbol{q})}{\partial \boldsymbol{q_a}} \end{array} \right]}_{\boldsymbol{J}(\boldsymbol{q})} \underbrace{\left[ \begin{array}{c} \dot{\boldsymbol{q}}_{\boldsymbol{h}} \\ \dot{\boldsymbol{q}}_{\boldsymbol{a}} \end{array} \right]}_{\dot{\boldsymbol{q}}}. \tag{1}$$

The terms $\frac{\partial \boldsymbol{f_\xi}(\boldsymbol{q})}{\partial \boldsymbol{q_h}}$ and $\frac{\partial \boldsymbol{f_\xi}(\boldsymbol{q})}{\partial \boldsymbol{q_a}}$ define two Jacobian matrices: $\boldsymbol{J_{\xi,h}}(\boldsymbol{q})$, which relates $\dot{\boldsymbol{\xi}}$ to the head velocity, and $\boldsymbol{J_{\xi,a}}(\boldsymbol{q})$, which relates $\dot{\boldsymbol{\xi}}$ to the arm velocity. Thus (1) can be rewritten

$$\dot{\boldsymbol{\xi}} = \boldsymbol{J_{\xi,h}}(\boldsymbol{q})\dot{\boldsymbol{q}}_{\boldsymbol{h}} + \boldsymbol{J_{\xi,a}}(\boldsymbol{q})\dot{\boldsymbol{q}}_{\boldsymbol{a}}. \tag{2}$$

In order to keep the target $\boldsymbol{\xi}^\star$ visible by the cameras of the robot, we choose to place it along the line of sight and reach it with the end effector. For that purpose we need to control the head and the arm independently from one another. Since we only control the head pitch rate $\dot{q}_{h_p}$ and yaw rate $\dot{q}_{h_y}$, we can implement the strategy of [7] to control the head in order to place the target along the line of sight

$$\begin{cases} \dot{q}_{h_p}^\star &= K_{h_p}\xi_y^\star + K_{D_{h_p}}\dot{\xi}_y^\star \\ \dot{q}_{h_y}^\star &= K_{h_y}\xi_x^\star + K_{D_{h_y}}\dot{\xi}_x^\star \end{cases}, \tag{3}$$

which leads asymptotically to $\xi_x^\star \to 0$ and $\xi_y^\star \to 0$.

Defining $\dot{\boldsymbol{q}}_{\boldsymbol{h}}^\star = \left(\dot{q}_{h_p}^\star, \dot{q}_{h_y}^\star\right)^T$ and $\dot{\boldsymbol{\xi}}_{\boldsymbol{a}}^\star = \dot{\boldsymbol{\xi}}^\star - \boldsymbol{J_{\xi,h}}(\boldsymbol{q})\dot{\boldsymbol{q}}_{\boldsymbol{h}}^\star$, the arm FVK (2) can be written

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{a}}^\star = \boldsymbol{J_{\xi,a}}(\boldsymbol{q})\dot{\boldsymbol{q}}_{\boldsymbol{a}}^\star, \tag{4}$$

where $\dot{\boldsymbol{q}}_{\boldsymbol{a}}^\star$ is the velocity to apply to the arm in order to exponentially decrease the feature errors given a head velocity $\dot{\boldsymbol{q}}_{\boldsymbol{h}}^\star$ used to place the target along the line of sight.

In the non singular case, this mapping is redundant [5] and there is an infinite number of solutions $\dot{\boldsymbol{q}}_{\boldsymbol{a}}^\star$ satisfying (4). Among them, the one provided by the Moore-Penrose pseudo-inverse $\boldsymbol{J_{\xi,a}}(\boldsymbol{q})^+$ provides minimum norm solutions (see [13]). To avoid critical effects due to singular configurations, the Damped Least Square Pseudo-inverse (DLS-PINV) of $\boldsymbol{J_{\xi,a}}\left(\boldsymbol{q}\right)$ is preferred. It can be computed based on the SVD of

$J_{\xi,a}(q)$ but an intuitive expression of the DLS-PINV is given by

$$J_{\xi,a}(q)^+ = J_{\xi,a}(q)^T \left( J_{\xi,a}(q) J_{\xi,a}(q)^T + k^2 I \right)^{-1} \quad (5)$$

where $I$ is the identity matrix. The parameter $k^2$ can be determined adaptively in different ways and we have chosen to use the method proposed in [14],

$$k^2 = \begin{cases} 0 & \sigma_{\min} > \bar{\sigma} \\ [1 - (\frac{\sigma_{\min}}{\bar{\sigma}})^2]k_{\max}^2 & \sigma_{\min} < \bar{\sigma} \end{cases} \quad (6)$$

where $\sigma_{\min}$ is the smallest singular value of the Jacobian matrix, $k_{\max}$ is a regularization term that we set to $0.316 \times (180/\pi)$ and $\bar{\sigma}$ is a threshold which determines if the plant is in singularity or not, that we set to $7.10^{-4}$. The obtained solution is not exact but is robust to singular configurations while ensuring minimal errors in any other case. Redundancy can also be used to partially achieve a secondary task while not disturbing the main one (in our case, reaching). This decoupling requires the computation of $P_{J_{\xi,a}}(q)$, the projector onto the null space of $J_{\xi,a}(q)$. It can be done using the SVD of $J_{\xi,a}$ and the column vectors $v_i$ of the resulting matrix $V$. As a consequence, the general form of the solution to (4) can be written

$$\dot{q}_a^\star = J_{\xi,a}(q)^+ \dot{\xi}_a^\star + P_{J_{\xi,a}}(q)\dot{q}_0, \quad (7)$$

where $\dot{q}_0 = -k_{\text{null}}\frac{\partial \mathcal{H}}{\partial q}$ is chosen accordingly to [15] in order to maintain the configuration far from joint limits ($k_{\text{null}} > 0$).

The use of the inverse velocity kinematics scheme described by (7) requires to have access to $J(q)$. The next subsections give a brief overview of the learning methods used in this work to incrementally approximate $J(q)$ based on joint positions and velocities as well as visual features measurements.

### D. LWPR *in a nutshell*

The Locally Weighted Projection Regression (LWPR) algorithm [16] is a recursive function approximator, which provides accurate approximation in very large spaces at low computational cost. It uses a sum of linear models weighted by normalized Gaussians. These Gaussians, also called receptive fields, define a zone of influence for each corresponding linear model. The receptive fields and the corresponding linear models are both updated incrementally to match the training data. LWPR reduces the input dimensionality using the Partial Least Squares (PLS) algorithm [17], [18], [19]. The global algorithm provides as output the weighted sum of all outputs of each receptive field

$$\hat{y}(x) = \frac{\sum_{k=1}^K w_k \hat{y}_k(x)}{\sum_{k=1}^K w_k} \quad (8)$$

where K is the number of receptive fields. We refer the reader to [20] or [21] for a presentation of the incremental version of the algorithm.

### E. XCSF *in another nutshell*

XCSF [22] is another function approximator that shares some similarities with LWPR but comes from Learning Classifier Systems (LCSs) [23]. As any LCS, XCSF manages a population of rules, called *classifiers*. These classifiers contain a *condition part* and a *prediction part*. In XCSF, the condition part defines the region of validity of a local model whereas the prediction part contains the local model itself.

A classifier defines a domain $\phi_i(z)$ and uses a corresponding linear model $\beta_i$ to predict a local output vector $y_i$ relative to an input vector $x_i$. The linear model is updated using the Recursive Least Squares (RLS) algorithm, the incremental version of the Least Squares method. The classifiers in XCSF form a population $P$ that clusters the condition space into a set of overlapping prediction models. XCSF uses only a subset of the classifiers to generate an approximation. Indeed, at each learning iteration, XCSF generates a match set $M$ that contains all classifiers in the population $P$ whose condition part $\mathcal{Z}$ matches the input data $z$ i.e., for which $\phi_i(z)$ is above a threshold $\phi_0$[4].

In XCSF, the output $\hat{y}$ is given for a $(x, z)$ pair as the sum of the linear models of each matching classifier $i$ weighted by its fitness $F_i$

$$\hat{y}(x, z) = \frac{\sum_{k=1}^{n_M} F_k(z) \hat{y}_k(x)}{\sum_{k=1}^{n_M} F_k(z)} \quad (9)$$

where $n_M$ is the number of classifiers in the match set $M$. In all other respects, the mechanisms that drive the evolution of the population of classifiers are directly inherited from XCS and are described in [24].

An important process in the context of this study is *compaction*. At the end of a learning process, the final population is composed of highly overlapping classifiers. To reduce the size of the population, XCSF uses a Closest Classifier Matching (CCM) rule to have a fixed size match set $M$ [24].

## III. EXPERIMENTS AND RESULTS

We use LWPR or XCSF to learn $J(q)$, the overall FVK expressed in the head frame of a ball handed by the robot with a stick. More specifically, the retained control scheme requires the access to $J_{\xi,h}(q)$ and $J_{\xi,a}(q)$ which are blocks of $J(q)$. As a matter of fact, we learn the FVK under a matrix form. Learning the FVK directly with LWPR would not provide such a matrix. Thus, instead, when using LWPR we learn $f_\xi(q)$ and use the first order derivative of the learned function (in our case, $J(q)$). Using XCSF, the Jacobian matrix is learned directly providing $q$ as condition parameter and $\left(\dot{q}, \dot{\xi}\right)$ as prediction parameter. The capacity to separate those spaces leads also to the possibility of predicting the learned model with only one condition parameter.

---

[4]This threshold is named $\theta_m$ in [24]

## A. Tuning the learning algorithms

LWPR with XCSF come with a large set of parameters and the performance is highly dependent on the effort put in tuning these parameters. To circumvent this difficulty, our methodology consists in the systematic exploration of these parameters with an equal amount of time attributed to tuning both algorithms. All parameters are tested on random joint positions and velocities as input. The output is computed using an existing KDL model as a baseline which provides a good estimation of the behavior of the system while speeding-up the exploration process. We consider having a sufficiently accurate prediction with $3.10^4$ training samples for the FVK.

The parameters tested with LWPR are described in the following table. All combinations of all values are tested.

| Tested parameters | values |
|---|---|
| init_d | 20, **25**, 30, 40, 50 |
| init_alpha | 0.01, 0.1, 1, 10, 100, **200**, 500 |
| w_gen | **0.01**, 0.1, 0.2 |
| penalty | 0.0001, 0.01, **0.1**, 0.5 |
| update_D | **true**, false |
| useMeta | **true**, false |

The parameters used for subsequent experiments are in bold face. They are chosen so that the NMSE converges to the lowest asymptotic value while decreasing fast in the first steps. The input dimension is 6 and the output dimension is 3. The norm_in parameter is set to 180. The corresponding performance evolution is shown in Fig. 2(a).

As XCSF may use different types of prediction space (quadratic or linear) and condition spaces (ellipsoid, rectangle, sphere, rotating or not), we choose linear predictions and rotating ellipsoid conditions in order to easily compare XCSF with LWPR.

The parameters tested with XCSF are described in the following table. As previously, the parameters which give the best values are in bold face.

| Tested parameters | values |
|---|---|
| maxPopSize | 500, 700, 1000, **1500**, 2000 |
| epsilon_0 | **0.01**, 0.001 |
| minConditionStretch | **0.005**, 0.001, 0.01 |
| coverConditionRange | 0.995, 0.7, **0.9** |
| delta | 0.1, 0.01, 0.05, **0.2**, 0.5 |
| startCompaction | 0.2, **0.4, 0.6** |
| doNumClosestMatch | **true**, false |

The condition and prediction input dimensions are set to 6. The output dimension is 3. The maximum learning iteration parameter is set to 150000. Since the error is approximately the same for startCompaction = 0.4 and 0.6, we finally choose startCompaction = 0.5. The corresponding performance evolution is shown in Fig. 2(b).

## B. Performing the asterisk experiment with the iCub simulator

In this experiment, we compare the results obtained with LWPR and XCSF in a reaching task. Results obtained with KDL are provided as a baseline. Our approach is tested on the 'star-like' asterisk task (see [25]) which consists in a go and return from a center point to eight target points that are visited sequentially in a clockwise manner. Unlike [25] where the whole trajectory is specified, only the target points are given. The center of the asterisk is placed in $\boldsymbol{\xi^*} = (\xi_x^* = 0.13m, \xi_y^* = 0.3m, \xi_z^* = 0.24m)^T$ relatively to the torso frame (cf. Fig. 1). The radius of the asterisk is $9cm$ and a target is considered to be reached under a threshold of $1cm$.

The closed loop control and learning implementation are described in Fig. 3. The learning algorithm module provides $J(\boldsymbol{q})$ after a learning period. The visualization module is a display using a virtual camera. The reprojection module computes the 3D position of the end effector with respect to the head. The control loop module computes the joint velocities needed to track the target with the head and the arm independently, as described in Section II-C.
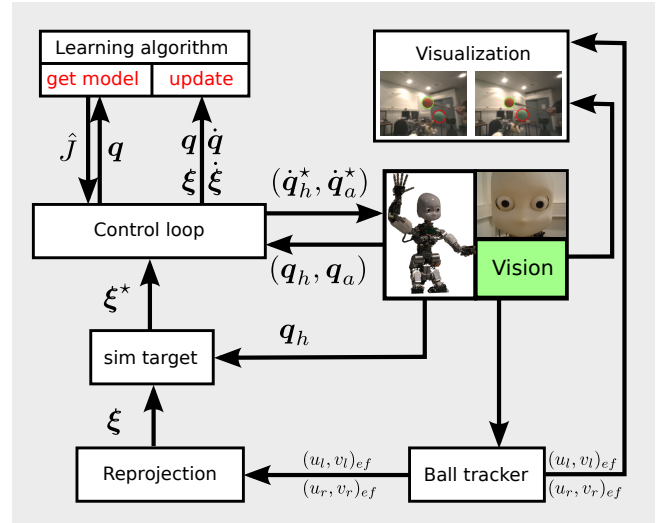


Fig. 3: On-line learning and closed-loop control. Each box is a YARP module

In Fig. 4, one can see the trajectory, in 2D, performed by the end effector during the first, 17th and 34th experiments, with the KDL model, LWPR, and XCSF, respectively. One can see that the trajectory does not change with the KDL model, apart from some minor variability due to control and vision noise. Moreover, the trajectory is initially worse with LWPR and XCSF, it improves faster with LWPR than with XCSF, but finally the XCSF model gets even more accurate after condensation. Those results are confirmed in Fig. 5, where one can see the evolution of the time necessary to perform a complete asterisk motion. Actually, the 34th asterisk is performed in 77 seconds with KDL, 71 with LWPR and 56 with XCSF.
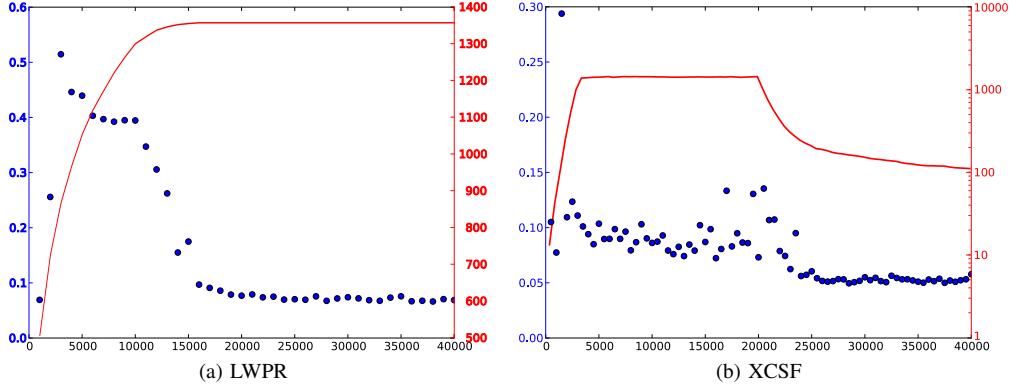
Fig. 2: Evolution of the normalized mean square error (left blue circle) and number of receptive fields (right red plain line) during 40000 learning steps with LWPR (a) and XCSF (b). With LWPR, the learning output is $\boldsymbol{\xi}$ whereas XCSF uses $\boldsymbol{q}$ as condition parameter and $\left(\dot{\boldsymbol{q}}, \dot{\boldsymbol{\xi}}\right)$ as prediction parameter.
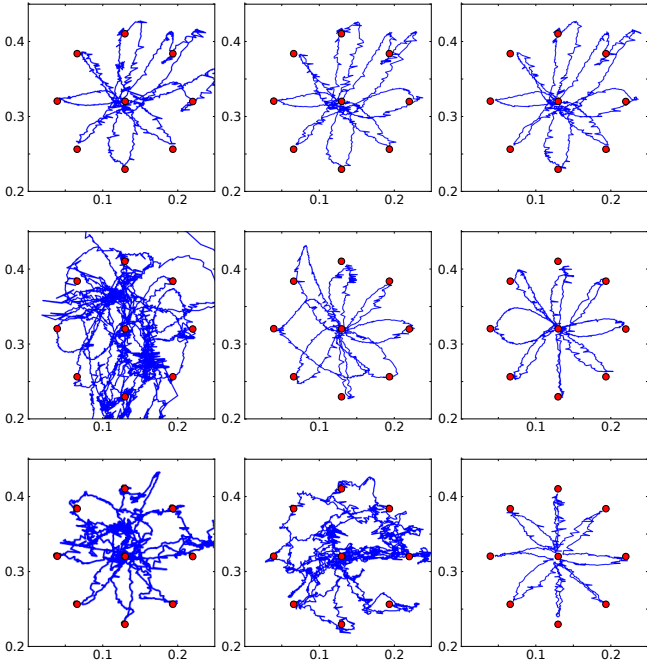


Fig. 4: Simulation results. First line: trajectory of the end effector using KDL. Second line: using LWPR. Third line: using XCSF. The columns represent the evolution between the first, the 17th and the 34th asterisk experiment respectively.
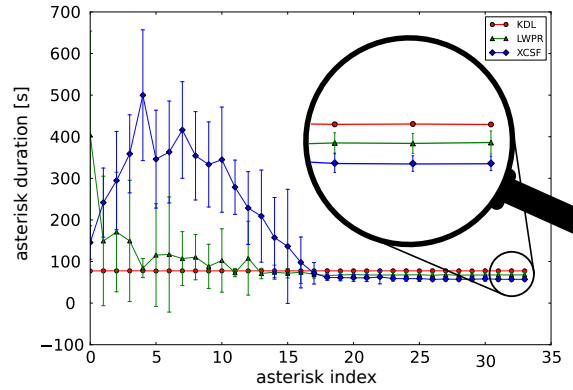


Fig. 5: Evolution of the time necessary to perform an entire asterisk task with the three used algorithms: KDL in red, LWPR in green and XCSF in blue.

## IV. DISCUSSION

First, independently from the model used for control, the reaching results and associated Cartesian trajectories are far from perfect. This is partly due to the fact that at the joint torque level of ICUB, control is performed in a decentralized manner: the whole-body dynamics is not accounted for and torque at joint $i$ is computed using a PID like control structure only relying on the error in velocity at joint $i$.

Moreover, it is often objected to the use of machine learning methods for FVK identification that it is easy to get this model analytically from the CAD model, which is itself generally very accurate. Our experiments show that the controllers using the FVK learned from both LWPR and XCSF outperform the one relying on the KDL model. This surprising result is explained by three factors. First, the control loop relies on visual information that is slightly wrong mainly because of poor calibration of the cameras of the robot. Indeed, in the case of ICUB, the origin of the visual frame of reference is not very precisely specified. Second, a detailed analysis of the KDL model revealed that it is slightly erroneous due to the evolution of the prototype over the years[5]. Third, in the case of real robot experiments, the position of the stick in the hand of the robot may not be precisely specified and may change from an experiment to another, which may induce errors in the estimation of the ball position relatively to the hand. The ball

---

[5]The model has been updated shortly after the realization of the research described here.

can be considered as a generic tool which is usually difficult to model accurately in the FVK of the robot. Learning algorithms can compensate for to all those uncertainties.

Finally, XCSF slightly outperforms LWPR on the tasks tested in this paper. Furthermore, as Fig. 2 shows, the final model is about ten times smaller with XCSF than with LWPR (about 100 classifiers against about 1350 receptive fields). LWPR is one of the most used machine learning methods. It has been evaluated on higher dimensional mechanical systems than XCSF, but always learning the model only along specific trajectories [26], whereas here we learn over the whole joint space of the studied system. Recently, [27] concluded that XCSF was outperforming LWPR on simple function approximation problems. Our work shows that this is also true for learning the FVK of the upper body of ICUB under realistic simulation conditions, which is a more significant result. Nevertheless, a gap exists between the simulated and the real robot and optimal parameters obtained in simulation may not be optimal for real world experiments, thus we must now perform evaluations on the real robot.

## V. CONCLUSION AND PERSPECTIVES

In this paper we have evaluated the applicability of a control framework based on visual servoing and model learning to the problem of controlling the ICUB humanoid robot. More precisely, we have compared the performance of LWPR and XCSF, showing that both were able to learn a model that is appropriate for control, with a slightly better performance and a much smaller model for XCSF.

Preliminary trials have shown that the approach can be transferred to the real robot, despite a greater variability due to additional sources of noise. In our immediate research agenda, we have to quantitatively evaluate the results on the real robot. Then we want to show that our approach to visual servoing with learning a separate model for the head and for the arm can be exploited to deal with the case where the robot is seeing the target, but not its arm. On a longer term, we will make profit of the access to the dynamics resulting from the availability of the IDYN library [28] to try to learn the dynamics model of the robot in the context of interaction with objects.

## REFERENCES

[1] O. Sigaud and J. Peters, "From motor learning to interaction learning in robots," in *From Motor Learning to Interaction Learning in Robots*, O. Sigaud and J. Peters, Eds. Springer, 2010, ch. 1, pp. 1–12.

[2] S. Vijayakumar, A. D'Souza, and S. Schaal, "LWPR: A scalable method for incremental online learning in high dimensions," Edinburgh University Press, Tech. Rep., 2005.

[3] J. Peters and S. Schaal, "Learning to control in operational space," *International Journal in Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.

[4] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Real-time local gaussian processes model learning," in *From Motor to Interaction Learning in Robots*, O. Sigaud and J. Peters, Eds. Springer, 2010.

[5] C. Salaün, V. Padois, and O. Sigaud, "Control of redundant robots using learned models: an operational space control approach," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, oct 2009, pp. 878–885.

[6] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The iCub humanoid robot: an open platform for research in embodied cognition," in *Permis: performance metrics for intelligent systems workshop*, Washington DC, USA, Aug. 2008.

[7] L. Natale, F. Nori, G. Metta, and G. Sandini, "Learning precise 3d reaching in a humanoid robot," in *Proceedings of the IEEE International Conference of Development and Learning (ICDL)*, London, UK, July 2007, pp. 1–6.

[8] M. V. Butz, D. Goldberg, and P. Lanzi, "Computational Complexity of the XCS Classifier System," *Foundations of Learning Classifier Systems*, vol. 51, pp. 91–125, 2005.

[9] V. Tikhanoff, P. Fitzpatrick, F. Nori, L. Natale, G. Metta, and A. Cangelosi, "The icub humanoid robot simulator," in *IROS Workshop on Robot Simulators*, 2008.

[10] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 29–45, 2008.

[11] R. Smits, T. De Laet, K. Claes, P. Soetens, J. De Schutter, and H. Bruyninckx, "Orocos: A software framework for complex sensordriven robot tasks," *IEEE Robotics and Automation Magazine*, 2008.

[12] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[13] K. Doty, C. Melchiorri, and C. Bonivento, "A theory of generalized inverses applied to Robotics," *The International Journal of Robotics Research*, vol. 12, no. 1, pp. 1–19, Feb. 1993.

[14] S. Chiaverini, O. Egeland, and R. K. Kanestrom, "Achieving user-defined accuracy with damped least-squares inverse kinematics," in *Proc. Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments', 91 ICAR*, June 1991, pp. 672–677.

[15] E. Marchand, F. Chaumette, and A. Rizzo, "Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing," in *Intelligent Robots and Systems, 1996 IEEE/RSJ International Conference on*, 1996.

[16] Statistical Machine Learning and Motor Control Group, "Locally Weighted Projection Regression," http://www.ipab.informatics.ed.ac.uk/slmc/software/lwpr, 2009.

[17] H. Wold, "Soft modelling by latent variables: the non-linear iterative partial least squares (NIPALS) approach," *Perspectives in Probability and Statistics (papers in honour of MS Bartlett on the occasion of his 65th birthday)*, pp. 117–142, 1975.

[18] L. Elden, "Partial least-squares vs. Lanczos bidiagonalization-I: analysis of a projection method for multiple regression," *Computational Statistics and Data Analysis*, vol. 46, no. 1, pp. 11–31, 2004.

[19] M. Tenenhaus, *La régression PLS: théorie et pratique.* Editions Technip, 1998.

[20] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[21] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford, CA*, 2000, pp. 1079–1086.

[22] S. W. Wilson, "Classifiers that Approximate Functions," *Natural Computing*, vol. 1, no. 2-3, pp. 211–234, 2002.

[23] O. Sigaud and S. W. Wilson, "Learning classifier systems: A survey," *Journal of Soft Computing*, vol. 11, no. 11, pp. 1065–1078, 2007.

[24] M. V. Butz and O. Herbort, "Context-dependent predictions and cognitive arm control with XCSF," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation.* ACM New York, NY, USA, 2008, pp. 1357–1364.

[25] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[26] O. Sigaud, C. Salaün, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: a survey," *Robotics and Autonomous Systems*, vol. 59, pp. 1115–1129, 2011.

[27] P. Stalph, J. Rubinsztajn, O. Sigaud, and M. Butz, "A Comparative Study: Function Approximation with LWPR and XCSF," in *Proceedings of the 13th International Workshop on Advances in Learning Classifier Systems*, 2010.

[28] S. Ivaldi, M. Fumagalli, M. Randazzo, F. Nori, G. Metta, and G. Sandini, "Computing robot internal/external wrenches by means of inertial, tactile and f/t sensors: theory and implementation on the icub," in *Proc. of the 11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, 2011.