Autonomous Online Learning of Velocity Kinematics on the iCub: a Comparative Study

Alain Droniou, Serena Ivaldi, Vincent Padois, Olivier Sigaud

Autonomous Online Learning of Velocity Kinematics on the iCub: a Comparative Study

Alain Droniou, Serena Ivaldi, Vincent Padois and Olivier Sigaud

***Abstract*— In the last years, several regression algorithms have been proposed to learn accurate mechanical models of robots. Comparisons are proposed at the conceptual level or through the use of recorded databases, but they deliver limited conclusions with respect to the real performance of these algorithms in their true context of use, i.e. online learning on the real robot interacting with its environment, within a feedback control loop. In this paper, we provide an empirical study of three state-of-the-art regression methods through online learning on the iCub robot holding a tool. We show that they can effectively learn a visuo-motor kinematic model for a simple visual servoing task in a very limited time (few minutes), without making any a priori hypothesis on the geometry of the robot and its tool. Furthermore, we can draw from the results some stronger conclusions about the comparison of the algorithms than previous studies based on databases.**

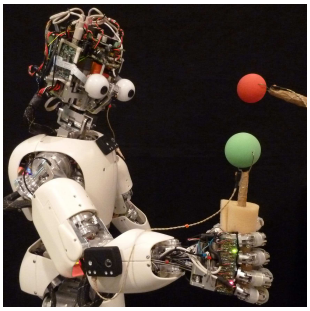


Fig. 1. Experimental setup: the humanoid robot iCub, holding an unknown tool, ideally represented by the green ball on the stick, reaching for a target, represented by the red ball. Motion capture markers used for recording the experiments are also visible.

I. INTRODUCTION

The increasing complexity of tasks addressed by humanoid robotics requires accurate geometrical and mechanical models which can be difficult to obtain in practice, especially when they are required to use a variety of tools. In such situations, robot controllers cannot purely rely on hard-coded models, even when precise CAD models of the robot are available [1]. The main reason for this is that kinematic controllers are not usually aware of the real dynamics of the robot. They rely on decentralized joint level controllers which account for rigid-body dynamics and non-linear effects as unknown disturbances and thus do not act as pure integrators. Also, they assume that geometrical/mechanical models do not change over time, which does not fit in a real-life scenario where the robot may change its kinematics properties[1], due to wear or malfunction. A possible strategy consists in letting the robot learn autonomously its own models, for example by interacting with the environment through its end-effectors (its hands in the simplest case), and exploiting a visual feedback to improve its accuracy while performing a task. Here we focus on the case where the geometrical properties of the tool and the environment are not known in advance.

All the authors are with: Université Pierre et Marie Curie, Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222, Pyramide Tour 55 - Boite Courrier 173, 4 Place Jussieu, 75252 Paris CEDEX 5, France. Contact: firstname.lastname@isir.upmc.fr

This work is supported by the French ANR program (ANR 2010 BLAN 0216 01 - http://macsi.isir.upmc.fr) and partially by the RTE company through its chair "Robotics Systems for field intervention in constrained environments" hold by Vincent Padois.

[1]For example, elastic tendons actuating the joints may loosen in time, inducing small errors in the kinematics models which would be generally difficult to detect and automatically compensate for.

In such contexts, Machine Learning (*ML*) techniques supersede classical approaches grounded on accurate exhaustive modeling [2], [3], [4]. However, the application of such techniques to online learning embedded on a real robot performing a visuo-motor task are not so common. More importantly, performance comparisons of such techniques under these more challenging experimental conditions are very rare. Comparisons relying on recorded databases can only convey limited conclusions on the practical applicability of such techniques in real scenarios, because they do not address several additional difficulties: the potential danger of any inadequate physical movement from the robot (for itself and for the environment), the necessity to deal with collisions and visual occlusions, the presence of uncontrolled noise at different sensory and motor levels, the difficulty to perform reproducible experiments with an accurate timing, the limited time available for learning, etc.

In this paper we propose such a comparison in the context of the incremental and autonomous learning of the velocity kinematics of a humanoid robot. We show how it is possible to make the robot learn its visuo-motor velocity relationship for a visual servoing task in a limited time and with a final accuracy which outperforms the hard-coded model derived from the CAD description of the robot. Our approach stems from the analysis of [5] and [6]. The former explored the possibility of learning the velocity kinematics of a simulated robot, comparing two supervised *ML* algorithms, namely LWPR and XCSF. The latter added a third algorithm, ISSGPR, and performed a detailed comparison using a database recorded from a real robot performing a visual servoing task. Here we further investigate the performance of the algorithms in an on-line task where learning is performed from scratch.

In particular, we study to what extent conclusions drawn from recorded databases still hold in this setting. We also investigate the generalization capability of the three algorithms and evaluate the robustness of the parameter settings that we use to diverse learning contexts.

The paper is organized as follows. In Section II, we describe the visual servoing problem, and give a brief overview of the regression algorithms. In Section III, we describe the robotics setup and the design of experiments, focusing on the technical issues that have to be solved to perform an online comparison. In Section IV, we empirically compare the learning algorithms under several criteria: the time they need to learn a satisfactory model, the accuracy of that model, their generalization capability when evaluated in a different context and their robustness to parameter setting. Finally, in Section V, we highlight our main conclusions and sketch an agenda of future studies.

II. METHODS

In the paper we derive a closed-loop controller for a generic end-effector, whose inverse kinematics is learnt from experience, based on the visual feedback from the cameras of the robot. We denote by $\boldsymbol{\xi} \in \mathbb{R}^p$ the Cartesian pose of the end-effector expressed in a reference frame attached to the cameras (e.g. the head), and by $\boldsymbol{\xi}^\star \in \mathbb{R}^p$ the desired pose to attain. The controller drives the end-effector towards a desired target using $\dot{\boldsymbol{\xi}}^\star = \lambda(\boldsymbol{\xi}^\star - \boldsymbol{\xi})$, where $\lambda > 0$ is a positive definite matrix. Desired velocities $\dot{\boldsymbol{\xi}} \in \mathbb{R}^p$ in the Cartesian space are related to joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ through the Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{p \times n}$, s.t. $\dot{\boldsymbol{\xi}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$. When the kinematic chain is redundant with respect to the task, the joint velocities can be found using the Jacobian pseudo-inverse $\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}(\boldsymbol{q})\dot{\boldsymbol{\xi}} + \boldsymbol{P}_{J(\boldsymbol{q})}\dot{\boldsymbol{q}}_0$, where $\dot{\boldsymbol{q}}_0$ optimizes a secondary task projected, using $\boldsymbol{P}_{J(\boldsymbol{q})}$, in the null space of the Jacobian [7]. The Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$ is learnt incrementally and on-line. As a follow-up of the comparison presented in [6], the following regression algorithms are used through our experiments.

1) LWPR: The Locally Weighted Projection Regression (LWPR) algorithm is a recursive function approximator, using a sum of linear models weighted by normalized Gaussians, also called receptive fields. It usually provides accurate approximation along trajectories in very large spaces at low computational cost [8].

2) XCSF: XCSF is a function approximation algorithm based on Learning Classifier Systems [9]. XCSF manages a population of rules, called *classifiers*, which contain a *condition part* and a *prediction part*: precisely, the classifiers form a population that clusters the condition space into a set of overlapping prediction models [10].

3) ISSGPR: The Incremental Sparse Spectrum Gaussian Process for Regression (ISSGPR) is based on a regularized least square method with random features [11][2]. It can be seen as an approximation of the Fourier transform of the function to learn, which allows the exploitation of some regularity properties to make the learning process more robust and simpler. The quality of predictions depends on the number D of features which are the learned Fourier coefficients.

It is worth noticing that XCSF learns directly the velocity kinematics Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$ whereas LWPR and ISSGPR learn the forward kinematic model $\boldsymbol{\xi} = \boldsymbol{f}(\boldsymbol{q})$. Therefore, in their case the Jacobian is retrieved after derivation.

[2]ISSGPR is also referred as IRFRLS [6], [2]

III. ROBOTIC SETUP

Experiments are carried out on the iCub, a 104cm high full-body humanoid platform [12]. We use the head and the arm for a total of 6 DOF: pitch and yaw joints in the neck, the 3 DOF of the shoulder and the elbow. Thus $\boldsymbol{q} = [\boldsymbol{q_h}, \boldsymbol{q_a}]^\top$, with $\boldsymbol{q_h} \in \mathbb{R}^2$, $\boldsymbol{q_a} \in \mathbb{R}^4$. The kinematic chain is redundant with respect to the task space, which, considering only positions, is such that $\boldsymbol{\xi} \in \mathbb{R}^3$. As in [13], a gaze system ensures that the target is always visible by the robot cameras, controlling the head independently of the arm with a simple law: $\dot{\boldsymbol{q}}_h^\star = K\boldsymbol{\xi}^\star + K_D\dot{\boldsymbol{\xi}}^\star$, which asymptotically brings the error to zero. As in [5], [14], a Regularized Damped Least Squares Pseudo-Inverse is used to avoid singularities of the arm. At the beginning of the learning phase, the robot is exploring its space from scratch. Since the Jacobian $\boldsymbol{J}(\boldsymbol{q})$ is refined incrementally, the arm may accidentally collide with the robot's body or the environment during the learning phase. To prevent damages, a joint level impedance velocity controller is used: given a desired joint velocity $\dot{q}^\star$, the commanded torque τ at the joint is computed as the sum of an impedance term (with fixed stiffness κ_s and damping κ_d) and a gravity compensation torque τ_g. Precisely, $\tau = -\kappa_s(q - q^\star) - \kappa_d\dot{q}^\star + \tau_g$. The following stiffness-damping values are used for the arm joints: $(0.4, 0.03), (0.4, 0.03), (0.4, 0.03), (0.2, 0.01)$ $[Nm/deg, Nm/deg/s]$ for joints $0, 1, 2, 3$ respectively (shoulder and elbow). These values provide a compliant but not springy behavior during contacts (an evaluation of the stiffness-damping map for the iCub arm is presented in [15]). The feedforward torque τ_g is computed using the dynamics library iDyn and the CAD dynamical parameters of the iCub [16]. The end-effector and the target are identified through an OpenCV-based module in the camera images, providing their coordinates in the visual reference frames, $(u, v)_{\text{left}}, (u, v)_{\text{right}}$. Their Cartesian position in the 3D space is then estimated through a stereo reprojection module, based on a calibration procedure of the cameras. A graphical representation of the interconnection between the software modules is shown in Fig. 2.

A. Experimental tasks

We consider three online visual servoing experiments (see Fig. 3), where the robot is required to reach or track a

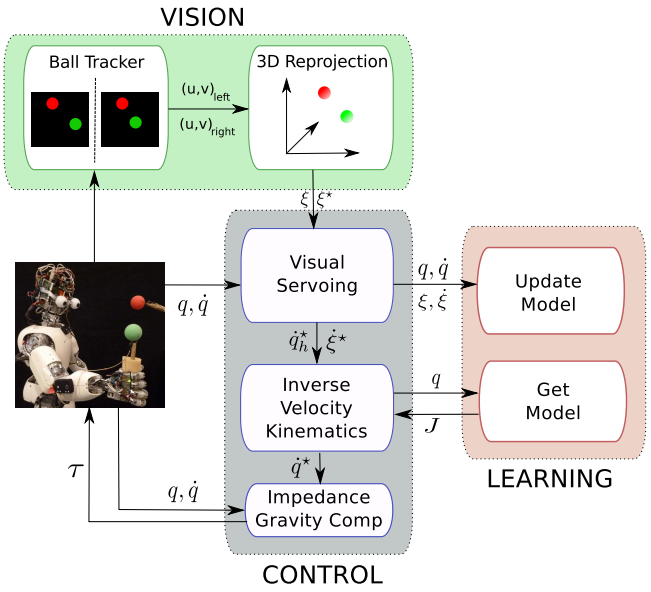
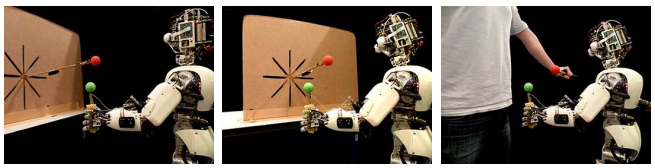


Fig. 2. Software architecture used for the online learning experiments.



(a) (b) (c)

Fig. 3. Three contexts for performing the experiments: the first two pictures show the exploration of two different workspaces during the asterisk experiments, while the third shows the robot interacting with a human partner for tracking the target moved randomly in the workspace.

target with the tip of a tool. To generalize over the end-effector and the target, we use a green and a red ball of $4cm$ diameter. The green ball is placed on top of a stick, held by the robot: this emulates an unknown tool used by the robot as end-effector. We consider a reaching movement successful when both balls collide. In [5], only the centers of the balls are considered, and they are emulated in the simulated environment. In [17], a minimum jerk trajectory between the centers is the reference for a tracking task, but is hard-coded in the controller.

The first two experiments consist in performing multiple reaching point-to-point trajectories, reproducing a star-like asterisk on a vertical plane. Eight target points are visited sequentially in a clockwise manner, first pointing outwards from a centered point then inwards back to the center. To improve reproducibility of "the asterisk" task, a rigid guide of about $32 \times 34cm$ is used to constrain the target to move in a vertical plane. In context A (see Fig. 3(a)), the plane is centered in front of the robot at about $30cm$. In context B (see Fig. 3(b)), the plane is rotated of 45 degrees and shifted on the right side of the robot, so that the distance between the target and the head varies between 10 and $35cm$, thus allowing the exploration of a different space. It must be noted that the target moves between the points, but is manually guided by a human operator holding a stick with a red ball. This induces a perturbation to the ideal nominal trajectory, that is straight lines between the points. This approach is necessary since the learning process may come with different solutions to drive the end-effector towards the target, with variable orientations, velocities etc. which are not predictable and could damage the robot with rigid setups even if it is compliant at joint level.

In the last experiment (context C - see Fig. 3(c)) the target is moved by the human in an unpredictable manner across the robot's workspace. This task allows an unconstrained exploration of the reachable space of the robot, at the price of a lesser reproducibility.

B. Performance metrics

In order to compare the performance of the learnt model, we track the Cartesian position of both end-effector and target through a CodaMotion motion capture system with active markers[3]. We interpolate the position of the center of the balls from two markers attached to the sticks, since markers directly placed on the balls can be occluded during contacts and obviously they cannot be placed precisely at the center of the ball. This solution induces inaccuracies in the estimation of the center of the balls. Since the latter are distant $4cm$ from each other when the balls are in contact, there is a bias in the evaluation of the recorded trajectory.

A precise quantification of the performance of the task is difficult to obtain in practice. In [5], the main performance metrics is the time necessary to draw an entire asterisk (globally 16 point-to-point movements) in the space, but the study is performed in a simulated environment. The target is instantaneously switching from one desired position to the next, there is no visual occlusion between the hand of the robot and the target, and there is no physical interaction between the end-effector and the target. In our study, all these impairments are met. The green ball (the end-effector) must physically touch the red one (the target); the latter moves from one desired position to the next along a trajectory which is not always perfectly straight in the Cartesian space, since it must avoid to collide with the arm when it is on the way. Moreover, during the exploration phase, it is not infrequent that the hand of the robot occludes the target in the visual field, which triggers an automated reset of the gaze controller into a predefined pose of the head, in search of the target. A similar mechanism is also used whenever the robot is not able to detect its end-effector, but in this case a coupled "head-arm reflex" is implemented, which drives both into an initial position. A similar mechanism is implemented in [18], inspired by [19]. All these elements cause the duration of each asterisk to be variable in practice with respect to a nominal performance, since in each trial minor changes in the learnt model can trigger one or more undesirable effects.

[3] www.codamotion.com

C. Parameters of the algorithms

In [6] a comparison among the aforementioned regression algorithms is presented, with an accurate statistical analysis of the influence of the parameters on the learning process. A database recorded from the iCub is used. To summarize, ISSGPR outperforms XCSF and LWPR in terms of both speed of convergence and precision. However, the number of features of ISSGPR has to be suitably chosen in order to ensure the responsiveness in real-time control, according to an accuracy/time trade-off. From the same database, we extract a set of optimal parameters for each algorithm for the asterisk tasks, which are reported hereinafter.

OPTIMIZED PARAMETERS FOR XCSF, LWPR AND ISSGPR

ISSGPR	
Parameter	Value
γ	10^{-6}
λ	10^{-12}
D	1000

XCSF		LWPR	
Parameter	Value	Parameter	Value
Δ	0.1	*useMeta*	true
converConditionRange	0.995	*updateD*	true
minConditionStretch	0.005	*penalty*	0.001
ϵ_0	0.005	*penalty*	0.2
maxPopulationSize	1500	*setInitAlpha*	0.01
startCondensation	500	*setInitD*	50

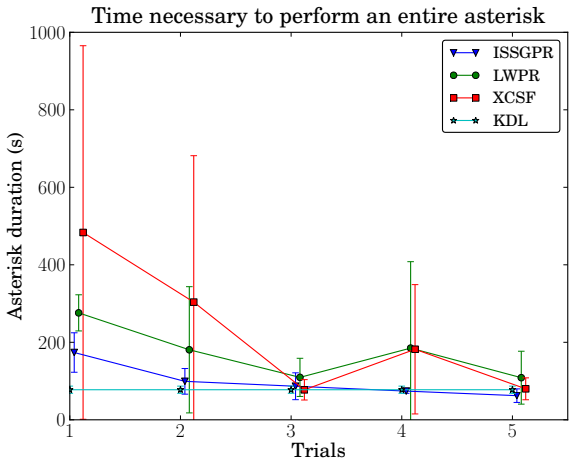


Fig. 4. Time necessary to perform an entire asterisk, trial after trial, for LWPR, XCSF, ISSGPR and using the KDL model.

IV. EXPERIMENTAL RESULTS

A. Learning from scratch

Once a suitable tuning of the parameters is found, as described in Section III-C, the set of optimal parameters can be used for learning from scratch, i.e. without bootstrapping the learning session with a local, demonstrated or initial model. The asterisks experiments are performed in context A (see Fig. 3(a)). As in [5], the performance metrics is the asterisk duration, i.e. the time necessary to execute an entire task. This performance is recorded after each new asterisk. Fig. 4 shows the comparison of the performance of the three algorithms while executing the asterisk task, averaged over four learning sessions. The performance of the KDL[4] model, i.e. the CAD model provided with the iCub software libraries, is also shown. Of course, this latter performance does not change since this model does not improve over time. On average, the time necessary to perform an entire asterisk task gradually decreases over learning. ISSGPR improves its performance faster than LWPR and XCSF, and finally performs better than its competitors and the KDL model. Fig. 5 shows the corresponding trajectory after the fifth asterisk. Due to many noise and perturbation phenomena that cannot be prevented in robotics experiments on a non industrial and complex robot such as iCub, these trajectories are still far from straight, even with the KDL model.

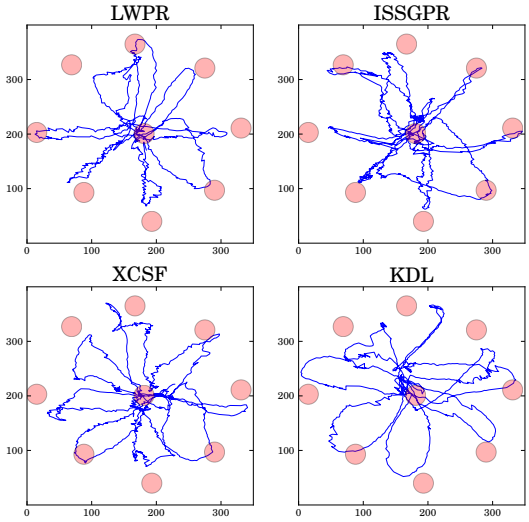


Fig. 5. Trajectory of the 5th asterisk of the learning process using LWPR, XCSF, ISSGPR and the KDL model (used as baseline). The red dots represent the target points and have approximately the same size than the ball.

B. Generalization capabilities

In order to highlight the generalization capabilities of the regression algorithms, we carry out the following experiment. We perform a learning session of each algorithm for precisely 10 minutes, where the task is to track the target along the asterisk in context A. Then a simple control session is performed in context B (see Fig. 3(b)) for 10 more minutes exploiting as such (i.e., learning is disabled) the model previously learnt in context A. Experimental results are shown in Fig. 6. As expected, LWPR and XCSF fail, since their generalization capability is very limited due to the use of many local models. Conversely, ISSGPR is able to perform the task in the "unknown" space. Since it learns the coefficients of the Fourier approximation of the model, it is

[4]www.orocos.org/kdl

less sensitive to the locality of the training data. Remarkably, this property allows also performing fast tracking in unexplored spaces, as described in context C. Fig. 8 show the performance of ISSGPR in tracking a target moved manually by the human at different velocities: since the human moves the target unpredictably, the explored space during tracking is different from the ones used for the learning. Besides a small delay (mostly due to vision processing) the tracking is quite fast and precise, especially given the reduced training before (less than 10 minutes).

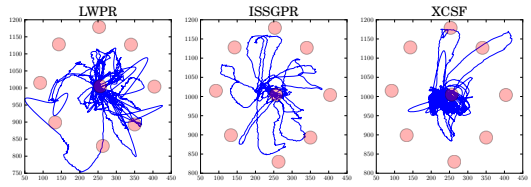


Fig. 6. Generalization capability of the three algorithms. Performance of a simple controller in context B, exploiting the models obtained after 10 minutes of a learning session in context A. Only ISSGPR manages to perform an entire asterisk (it performed 5 sequences in 10 minutes), whereas XCSF and LWPR were not even able to finish one sequence. More precisely, XCSF executed only 4 reaching movements and LWPR only 1 out of 16.

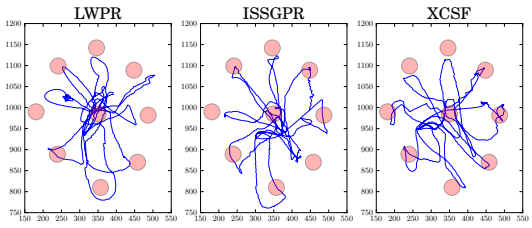


Fig. 7. Performance of the three algorithms when learning is performed in context B with parameters tuned for context A.

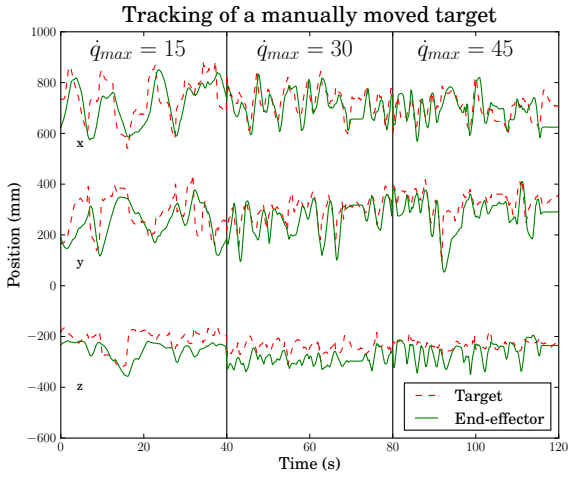


Fig. 8. Trajectories of the end-effector while tracking a target moved randomly by a human across the entire workspace, at different velocities. Only ISSGPR is shown.

C. Robustness of parameters

A general critique to the application of regression methods in robotics is that most of the state-of-the-art algorithms require a practical 'hands-on' expertise, especially in the choice of the numerous parameters affecting the performance of the learning process and in the amount of data which are required to attain a desired accuracy. The goal of the following experiment is to show that even if the parameters of the algorithms are tuned on a subset of the reachable workspace and are optimized for learning in that particular space, they are equivalently good for learning in another subset of the robot workspace. In other terms, we test the robustness of the learning algorithms with respect to the parameters which are tuned for a portion of the workspace which is different from the one explored during learning. The experiment consists in learning from scratch the visuo-motor Jacobian, performing the asterisk task in the space of context B with the parameters which are tuned for the context A. The former context is generally harder to explore: being closer to the robot, it exacerbates visual occlusions and contacts issues. Results are reported in Fig. 7. Overall, the performances of the learning algorithms are mildly affected by the context switch. This suggests that the parameters optimization does not necessarily requires a pervasive sampling of the state space explored during the online learning task.

D. Video

The effectiveness of the proposed approach is also evident from the video attached to the paper. The iCub performs the asterisk experiment and is able to incrementally learn its visuo-motor kinematics relationship with the three algorithms in a visual servoing task: the video shows a comparison among three sessions (with ISSGPR, LWPR and XCSF respectively) where learning is from scratch. The time to perform the first asterisk is coherent with the results shown in Fig. 4. For the selected trials, we have $140, 190, 280$ seconds respectively. ISSGPR performs better than the alternative *ML* algorithms: remarkably, only few seconds are necessary to have a model sufficiently good to track the desired target with roughly straight paths. We also show its performance in tracking a target moved manually by the human at different velocities, as in Fig. 8. The full video is available at `http://macsi.isir.upmc.fr`.

V. CONCLUSIONS AND FUTURE WORK

With a suitable tuning, state-of-the-art *ML* algorithms can be readily used in combination with closed-loop controllers in online robotics experiments. In this work, we show that after a short training session (e.g. five asterisks, corresponding to less than 10 minutes on average), the learnt velocity kinematics provided by regression algorithms performs comparably with the one based on the mechanical model of the robot. A longer training session may further improve the performance, as suggested in [5] for simulated

data. The numerous sources of perturbations during the execution of the trajectories induce a lot of noise which makes the comparison harder, and put additional complications to provide statistical significance tests, as is done in [6] on a recorded database.

An interesting point raised by this work is that comparing *ML* algorithms for obtaining mechanical models of a robot through online interaction experiments may not be as straightforward as on databases of recorded data. Indeed, similar works [20] circumvent some experimental issues by simulating the target to track in the operational space. However such approaches require to hard-code in the architecture the knowledge of the target trajectory. Thus a kinematics model of the robot is necessary to project the target in its workspace, which biases the performance analysis. Conversely, in our approach the learning process relies on a pure visual feedback and a physical interaction between end-effector and target. Occlusions and collisions cannot be avoided, and limit the repeatability of the trials. We plan to improve the experimental protocol to avoid these issues. First, we will implement predictive models which can avoid resetting both arm and head once the end-effector or the target are occluded or disappear from the visual field. Then, we will add a mechanism to better handle self-body contacts and generate escape trajectories when the robot collides with the environment.

The supplemental video attached to the paper shows the difficulties of the experimental setup, but also highlights the fast convergence and the performance of the algorithms in an online control task. In accordance with the results of [6], ISSGPR performs better than LWPR and XCSF with respect to all the criteria studied here. Indeed, when learning from scratch, ISSGPR is the fastest in terms of time of convergence and the most accurate in terms of function approximation. It also benefits from a far better generalization capability over a different workspace and it is not too sensitive to the set of parameters used to perform the experiments.

Future studies will investigate whether ISSGPR still outperforms when the dimensionality of the problem increases (both in terms of complexity of the task and number of controlled DOF in the robot). With the use of NIPALS as a regression method [21], we expect LWPR to learn more compact models that may make a difference when addressing whole body control.

Given the encouraging outcomes of our empirical studies, we plan to perform similar experiments for learning the robot dynamics during interactions with the environment. We will thereby confront the algorithms to less regular functions. Previous works in simulation [22] and with recorded databases [23], [24] provide a baseline for these investigations. However since the iCub is only endowed with proximal force/torque sensors and not with joint torque sensors, additional challenges in the definition of a suitable experimental protocol will be faced.

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] 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.

[3] M. Hoffmann, H. Marques, A. Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer, "Body schema in robotics: A review," *IEEE Trans. Autonomous Mental Development*, vol. 2, no. 4, pp. 304–324, 2010.

[4] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, pp. 1–22, 2011.

[5] G. Sicard, C. Salaün, S. Ivaldi, V. Padois, and O. Sigaud, "Learning the velocity kinematics of iCub for model-based control: XCSF versus LWPR," in *Int. Conf. on Humanoid Robots*, 2011, pp. 570 – 575.

[6] A. Droniou, S. Ivaldi, P. Stalph, M. Butz, and O. Sigaud, "Learning velocity kinematics: Experimental comparison of on-line regression algorithms," in *Robotica*, 2012, pp. 15–20.

[7] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed. Springer, 2005.

[8] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space," in *Int. Conf. on Machine Learning*, 2000.

[9] S. W. Wilson, "Function approximation with a classifier system," *Genetic and Evolut. Comp. Conf.*, pp. 974–981, 2001.

[10] M. V. Butz, P. L. Lanzi, and S. W. Wilson, "Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction," *IEEE Trans. on Evolutionary Computation*, pp. 12:355–376, 2008.

[11] A. Gijsbert, "Incremental learning for robotics with constant update complexity," Ph.D. dissertation, Italian Institute of Technology, 2011.

[12] L. Natale, F. Nori, G. Metta, M. Fumagalli, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz, and G. G. Sandini, *Intrinsically motivated learning in natural and artificial systems*. Springer-Verlag, 2012, ch. The iCub platform: a tool for studying intrinsically motivated learning.

[13] L. Natale, F. Nori, G. Metta, and G. Sandini, "Learning precise 3D reaching in a humanoid robot," in *Int. Conf. of Development and Learning*, 2007, pp. 1–6.

[14] S. Chiaverini, O. Egeland, and R. K. Kanestrom, "Achieving user-defined accuracy with damped least-squares inverse kinematics," in *Int. Conf. Advanced Robotics - ICAR*, 1991, pp. 672–677.

[15] B. Berret, S. Ivaldi, F. Nori, and G. Sandini, "Stochastic optimal control with variable impedance manipulators in presence of uncertainties and delayed feedback," in *Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 4354–4359.

[16] 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 *Int. Conf. on Humanoid Robots*, 2011, pp. 521–528.

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

[18] A. Baranes and P.-Y. Oudeyer, "Intrinsically motivated goal exploration for active motor learning in robots: a case study," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[19] N. Berthier, D. Mccall, R. Clifton, and D. Robin, "Proximodistal structure of early reaching in human infants," *Experimental Brain Research*, vol. 127, pp. 259–269, 1999.

[20] B. Bocsi, P. Hennig, L. Csato, and J. Peters, "Learning tracking control with forward models," in *Int. Conf. on Robotics and Automation*, 2012.

[21] H. Wold, "Soft modelling by latent variables: the non-linear iterative partial least squares (NIPALS) approach," *Perspectives in Probability and Statistics*, pp. 117–142, 1975.

[22] J. S. de la Cruz, D. Kulic, and W. Owen, "Online incremental learning of inverse dynamics incorporating prior knowledge," in *Int. Conf. on Autonomous and Intelligent Systems*, 2011.

[23] A. Gijsberts and G. Metta, "Incremental learning of robot dynamics using random features," in *Int. Conf. on Robotics and Automation*, 2011, pp. 951–956.

[24] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *Int. Conf. on Robotics and Automation*, 2010.