

Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations

Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude Billard
Learning Algorithms and Systems Laboratory - LASA
School of Engineering - EPFL
Station 9 - 1015 Lausanne - Switzerland

Abstract

We present a system for robust robot skill acquisition from kinesthetic demonstrations. This system allows a robot to learn a simple goal-directed gesture, and correctly reproduce it despite changes in the initial conditions, and perturbations in the environment. It combines a dynamical system control approach with tools of statistical learning theory and provides a solution to the inverse kinematics problem, when dealing with a redundant manipulator. The system is validated on two experiments involving a humanoid robot: putting an object into a box, and reaching for and grasping an object.

Index Terms

Robot Programming by Demonstration, Dynamical System Control, Gaussian Mixture Regression

Corresponding author: Micha Hersch, micha.hersch@epfl.ch
Conditionally accepted short paper submitted to the IEEE Transactions on Robotics
Paper # A07-471/A06-417

Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations

I. INTRODUCTION

AS robots are progressively coming out of the controlled environment of assembly lines to pervade the much less predictable domestic environments, there is a need to develop new kinds of controllers that can cope with changing environments and that can be taught by unskilled human users. In order to address this last issue, *Programming by Demonstration* (PbD) has emerged as a promising approach [1]. PbD has been mostly used in two cases: for tasks involving no or very loose interaction with the environment (like writing, martial arts or communicative gestures) human demonstrations are used to train a movement model, which can be used to reproduce the task. Those movement models (also used in computer animation or visual gesture recognition) usually imply some averaging process (LWR [2], HSTMM [3]), possibly in a latent space (GPLVM [4], ST-Isomap [5]) or some probabilistic model like Bayesian Networks [6]. And for more complex tasks, involving precise interactions with the environment, the robot learns from examples how to sequence a set of hard-coded controllers for a given task. This has been done using HMMs [7] or knowledge-based systems [8].

In our work, we position ourselves in between those two approaches and combine learning of a task-dependent modulation of a built-in controller. We, thus, start with a basic built-in controller (or motion primitive) that consists in a dynamical system with a single stable attractor. We modulate the trajectories generated by this controller to be task-specific, by learning a probabilistic model of the task-based trajectory, as shown by a human user. This results in a general framework for learning and reproducing goal-directed gestures, despite different initial conditions and changes occurring during task execution. In this respect, we improve in several ways classical control approaches for goal-directed motions, such as [9].

The closest work to ours is [2], which uses a dynamical system for goal-directed reaching. We depart from this work in two ways: First, we propose a hybrid controller composed of two of our basic dynamical systems working concurrently in end-effector and joint angle spaces. This results in a controller that has no singularities. Second, the dynamical system approach gives us a controller robust in the face of perturbations, which can recompute the trajectory on-line to adapt to sudden displacements of the target or unexpected motion of the arm during motion, and we provide experimental results on the robustness to static and dynamic changes in the environment. While our controller is less precise than ad-hoc controller (e.g. [10]), it is more general in that it can be easily modulated to achieve arbitrary goal-directed reaching tasks.

In the experiments presented here, the motions are demonstrated to the robot by a human user moving the robots' limbs

passively (kinesthetic training). In Section IV, we validate the approach on two different tasks, namely placing an object into a box, and reaching-to-grasp a chess piece, see Fig. 2 for illustrations of these two tasks.

II. OVERVIEW

The system is designed to enable a robot to learn to modulate its generic controller to produce any arbitrary goal-directed motion. The model must be generic so as to reproduce the motion given different initial conditions and under perturbations during execution. Moreover, the architecture of the system must permit the use of different control variables for encoding the motion. Here, we compare a control in either velocity or acceleration. We refer to those further as the *velocity model* (see Section II-B) and the *acceleration model* (see Section II-C).

A. System Architecture

The structure of the system is the same for both models and is schematized in Fig. 1. During training, the relevant variables (end-effector velocity profiles for the velocity model, or end-effector positions, velocities and accelerations for the acceleration model) are extracted from the set of demonstrated trajectories and used to train a Gaussian Mixture Model (GMM) (see Table I). During reproduction, the trajectory is specified by a spring-and-damper dynamical system modulated by the GMM (see section III). The target is tracked by a stereo-vision system and is set to be the attractor point of the dynamical system. At each time step, the desired velocity computed by the model is then fed to a PID controller for execution. This does not hinder the online adaptation of the movement.

B. Velocity Model

The first way to encode a motion in a GMM, is to consider the velocity profile of the end-effector as a function of time $\dot{\mathbf{x}}(t)$. Thus, the input variable ζ is the time and the output variable ξ is the velocity, like in the following velocity model:

$$\dot{\mathbf{x}}^m = \tilde{\mathcal{F}}_{\mathbf{x}}(t) \quad (2)$$

In other words, the movement is modeled as a velocity profile, given by a function of time, which is learned as described in Table I. Here and henceforth, $\dot{\mathbf{x}}^m \in \mathbb{R}^m$ is the end-effector velocity specified by the task model. $\tilde{\mathcal{F}}_{\mathbf{x}}$ is obtained by applying (1) with the appropriate variables.

TABLE I
SUMMARY OF GAUSSIAN MIXTURE REGRESSION (GMR).

GMR is a method suggested by [11] for statistically estimating a function $\mathcal{F}_\xi(\zeta)$ given by a “training set” of N examples $\{(\zeta^i, \xi^i)\}_{i=1}^N$, where ξ^i is a noisy measurement of $\mathcal{F}_\xi(\zeta^i)$:

$$\xi^i = \mathcal{F}_\xi(\zeta^i) + \epsilon^i$$

(ϵ^i is the Gaussian noise). The idea is to model the joint distribution of the “input” variable ζ and an “output” variable ξ as a Gaussian Mixture Model. If we join those variables in a vector $v = [\zeta^T \xi^T]^T$, it is possible to model its probability density function as a mixture of K Gaussian functions

$$p(v) = \sum_{k=1}^K \pi_k \mathcal{N}(v; \mu_k, \Sigma_k), \quad \text{such that} \quad \sum_{k=1}^K \pi_k = 1$$

where the $\pi_k \in [0, 1]$ are the priors, and $\mathcal{N}(v; \mu_k, \Sigma_k)$ is a Gaussian function with mean μ_k and covariance matrix Σ_k :

$$\mathcal{N}(v; \mu_k, \Sigma_k) = ((2\pi)^d |\Sigma_k|)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(v - \mu_k)^T \Sigma_k^{-1} (v - \mu_k)\right),$$

where d is the dimensionality of the vector v . The mean vectors μ_k and covariance matrices Σ_k can be separated into their respective input and output components:

$$\mu_k = [\mu_{k,\zeta}^T \mu_{k,\xi}^T]^T \quad \Sigma_k = \begin{pmatrix} \Sigma_{k,\zeta} & \Sigma_{k,\zeta\xi} \\ \Sigma_{k,\xi\zeta} & \Sigma_{k,\xi} \end{pmatrix}$$

The Gaussian Mixture Model (GMM) is trained using a standard E-M algorithm, taking the demonstrations as training data. The GMM computes a joint probability density function for the input and the output, so that the probability of the output conditioned on the input are GMM. Hence, it is possible, after training, to recover the expected output variable $\tilde{\xi}$, given the observed input variable ζ .

$$\tilde{\xi} = \tilde{\mathcal{F}}_\xi(\zeta) = \sum_{k=1}^K h_k(\zeta) (\mu_{k,\xi} + \Sigma_{k,\xi\zeta} \Sigma_{k,\zeta}^{-1} (\zeta - \mu_{k,\zeta})), \quad (1)$$

where the $h_k(\zeta)$ are given by:

$$h_k(\zeta) = \frac{\pi_k \mathcal{N}(\zeta; \mu_{k,\zeta}, \Sigma_{k,\zeta})}{\sum_{k=1}^K \pi_k \mathcal{N}(\zeta; \mu_{k,\zeta}, \Sigma_{k,\zeta})}.$$

The tilde (‘’) sign indicates that we are dealing with expectation values.

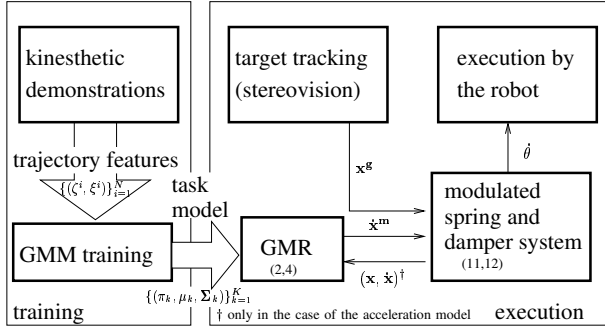


Fig. 1. The architecture of the system. During training the relevant variables (end-effector’s position, velocity and acceleration) are extracted from the demonstrations and used to train a GMM. During task execution, this model is used to modulate a spring-and-damper system. $\dot{\mathbf{x}}^m$ is the end-effector velocity specified by the task model. \mathbf{x}^s is the target location, and \mathbf{x}^* , $\dot{\mathbf{x}}^*$, $\dot{\theta}^*$ are respectively the actual current end-effector’s position and velocity and the joint angles’ velocities. The numbers in parentheses refer to the corresponding equations in the text.

C. Acceleration Model

A second way of encoding a trajectory is to take as input the position \mathbf{x} and velocity $\dot{\mathbf{x}}$, and as output the acceleration $\ddot{\mathbf{x}}$. The rationale of this is to consider a trajectory not as a function of time, but as the realization of a second-order dynamical system of the form:

$$\ddot{\mathbf{x}}^m = \tilde{\mathcal{F}}_{\ddot{\mathbf{x}}}(\mathbf{x}, \dot{\mathbf{x}}). \quad (3)$$

Again, $\tilde{\mathcal{F}}_{\ddot{\mathbf{x}}}$ is obtained by applying (1) with the appropriate variables. The velocity specified by the acceleration model is then given by

$$\dot{\mathbf{x}}^m = \dot{\mathbf{x}} + \tau \tilde{\mathcal{F}}_{\ddot{\mathbf{x}}}(\mathbf{x}, \dot{\mathbf{x}}), \quad (4)$$

where τ is the time integration constant (set to 1 in this paper). Since the position \mathbf{x} and velocity $\dot{\mathbf{x}}$ depend on the acceleration $\ddot{\mathbf{x}}$ at previous times, this representation introduces a feedback loop, which is not present in the representation given by (2).

III. MODULATED SPRING-AND-DAMPER SYSTEM

We now show how the task model described above is used to modulate a spring-and-damper dynamical system in order

to enable a (possibly redundant) robotic arm with n joints to reproduce the task with sufficient flexibility. Although the modulation $\dot{\mathbf{x}}^m$ is in end-effector space, it is advantageous (for avoiding singularity problems related to inverse kinematics of redundant manipulators) to consider the spring-and-damper dynamical system in joint angle variables:

$$\ddot{\theta}^s = \alpha(-\dot{\theta} + \beta(\theta^s - \theta)) \quad (5)$$

where $\theta \in \mathbb{R}^n$ is the vector of joint angles (or arm configuration vector). This dynamical system produces straight paths (in joint space) to the target θ^s , which acts as an attractor of the system. This guarantees that the robot reaches the target smoothly, despite possible perturbations.

The above dynamical system is modulated by the variable $\dot{\mathbf{x}}^m$ given by the task model (2) or (4). In order to weigh the modulation, we introduce a modulation factor $\gamma \in \mathbb{R}_{[0, 1]}$, which weighs the importance of the task model relatively to the spring-and-damper system. If $\gamma = 0$, only the spring-and-damper system is considered, and when $\gamma = 1$ only the task model is considered. In order to guarantee the convergence of the system to θ^s , γ has to tend to zero at the end of the movement. In the experiments described here, γ is given by:

$$\dot{\gamma} = \alpha_\gamma(-\dot{\gamma} - \frac{1}{4}\alpha_\gamma\gamma) \quad \text{with } \gamma_0 = 1, \quad (6)$$

where γ_0 is the initial value of γ and $\alpha_\gamma \in \mathbb{R}_{[0, 1]}$ is a scalar.

Since $\dot{\mathbf{x}}^m$ lives in the end-effector space (and not in the joint space), the modulation is performed by solving the following constrained optimization problem.

$$\begin{aligned} \dot{\theta} = \underset{\dot{\theta}}{\operatorname{argmin}} \quad & (1 - \gamma)(\dot{\theta} - \dot{\theta}^s)^T \bar{\mathbf{W}}_\theta (\dot{\theta} - \dot{\theta}^s) + \\ & \gamma(\dot{\mathbf{x}} - \dot{\mathbf{x}}^m)^T \bar{\mathbf{W}}_{\mathbf{x}} (\dot{\mathbf{x}} - \dot{\mathbf{x}}^m) \\ \text{u.c.} \quad & \dot{\mathbf{x}} = \mathbf{J}\dot{\theta}, \end{aligned} \quad (7)$$

where \mathbf{J} is the Jacobian of the robot arm kinematic function \mathbf{K} and $\bar{\mathbf{W}}_\theta \in \mathbb{R}^{n \times n}$ and $\bar{\mathbf{W}}_{\mathbf{x}} \in \mathbb{R}^{m \times m}$ are diagonal matrices necessary to compensate for the different scale of the \mathbf{x} and θ variables. As a rough approximation, the diagonal elements of $\bar{\mathbf{W}}_{\mathbf{x}}$ are set to one and those of $\bar{\mathbf{W}}_\theta$ are set to the average

distance between the robot base and its end-effector.

The solution to this minimization problem is given by [12]:

$$\dot{\theta} = (\mathbf{W}_\theta + \mathbf{J}^T \mathbf{W}_x \mathbf{J})^{-1} (\mathbf{W}_\theta \dot{\theta}^s + \mathbf{J}^T \mathbf{W}_x \dot{\mathbf{x}}^m) \quad (9)$$

$$\text{where } \mathbf{W}_\theta = (1 - \gamma) \bar{\mathbf{W}}_\theta, \quad \mathbf{W}_x = \gamma \bar{\mathbf{W}}_x. \quad (10)$$

To summarize, the task is performed by integrating the following dynamical system:

$$\ddot{\theta}^s = \alpha(-\dot{\theta} + \beta(\theta^g - \theta)) \quad (11)$$

$$\dot{\theta} = (\mathbf{W}_\theta + \mathbf{J}^T \mathbf{W}_x \mathbf{J})^{-1} (\mathbf{W}_\theta \dot{\theta}^s + \mathbf{J}^T \mathbf{W}_x \dot{\mathbf{x}}^m) \quad (12)$$

where \mathbf{W}_x and \mathbf{W}_θ are given by (6) and (10), and $\dot{\mathbf{x}}^m$ is given either by (2) (velocity model) or by (4) (acceleration model). Integration is performed using a first-order Newton approximation ($\dot{\theta}^s = \dot{\theta} + \tau \ddot{\theta}^s$).

Since the target location is given in cartesian coordinates, inverse kinematics must be performed in order to obtain the corresponding target joint angle configuration which will serve as input of the spring-and-damper dynamical system. In the case of a redundant manipulator (such as the robot arm used in the following experiments) the desired redundant parameters of the target joint angle configuration can be extracted from the demonstrations. This is done by using the GMR technique described in Table I to build a model of the final arm configuration as a function of the target location.

Using an attractor system in joint angle space has the practical advantage of reducing the usual problems related to end-effector control, such as joint limit and singularity avoidance. Equation 9, which is a generalized version of the Damped Least Squares inverse [13] [14], is a way to simultaneously control the joint angles and the end-effector, imposing soft constraints on both of them. It is thus different than optimizing the joint angles in the null space of the kinematic function.

IV. EXPERIMENTS

A. Setup

We validate and compare the systems described in this paper on two experiments. The first experiment involves a robot putting an object into a box and the second experiment consists in reaching and grasping for an object. Those experiments were chosen because (1) they can be considered as simple goal-directed tasks (for which the system is intended), (2) they are tasks commonly performed in human environments and (3) they presents a clear success or failure criterion.

All the experiments presented below are performed with a Hoap3 humanoid robot acquired from Fujitsu. This robot has four back-drivable degrees of freedom (dof) at each arm. Thus, the robot arms are redundant, as we do not consider end-effector orientation. The robot is endowed with a stereo-vision system enabling it to track color blobs. A small color patch is fixed on the box and on the object to be grasped, enabling their 3D localization. Pictures of the setup are shown in Fig. 2.

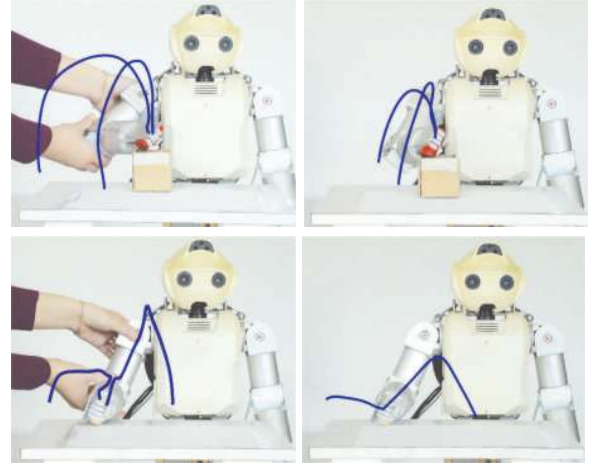


Fig. 2. The setup of the experiments. The top pictures show the first task and the lower picture show the second task Left: a human operator demonstrates a task to the robot by guiding its limbs. Right: the robot performs the task, starting from different initial positions.

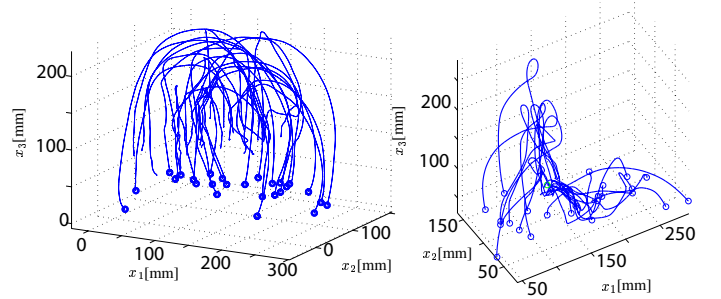


Fig. 3. The demonstrated trajectories for the box task (left) and the grasping task (right). Circles indicate starting positions.

1) *Preprocessing*: During the demonstrations, the robot joint angles were recorded and the end-effector positions were computed using the arm kinematic function. All recorded trajectories were linearly normalized in time ($T = 500$ time steps) and Gaussian-filtered to remove noise. The number of Gaussian components for the task models were found using the Bayesian Information Criteria (BIC) [15], and the parameter values used were $\alpha_\gamma = 0.06$, $\alpha = 0.12$ and $\beta = 0.06$.

B. Putting an object into a box

1) *Description*: For this task, the robot is taught to put an object into the box (see Fig.2). In order to accomplish the task, the robot has to avoid hitting the box while performing the movement and must thus first reach up above the box and then down to the box. A straight line reaching will in general cause the robot to hit the box while reaching and thus fail.

2) *Training*: A set of 26 kinesthetic demonstrations were performed, with different initial positions and box locations. The box was placed on a little table. Thus its location only varies in the horizontal plane. Similarly, the initial position of the object (and thus of the end-effector) lied on the table. The set of demonstrated trajectories is depicted in Fig. 3, left. The velocity models trained on this data are shown in Fig. 4, left.

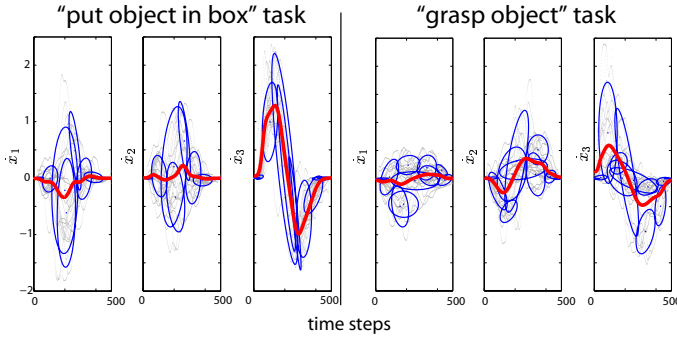


Fig. 4. The velocity models for both tasks. The dots represent the training data, the ellipses the Gaussian components and the thick lines the trajectory obtained by GMR alone. The thick lines show that, for the first task, the horizontal components \dot{x}_1 and \dot{x}_2 are averaged out by the model, but the vertical component \dot{x}_3 shows a marked upward movement. For the second task, all components are almost averaged out.

C. Reach and Grasp

1) *Description*: In order to accomplish this task, the robot has to reach and correctly place its hand to grasp a chess piece. In other words it has to place its hand so that the chess piece stands between its thumb and its remaining fingers, as shown in Fig. 7, left. This figure illustrates that the approaching the object can only be done in one of two directions: downward or forward. This task is more difficult than the previous one, as the movement is more constrained. Moreover, a higher precision is required on the final position, since the hand is relatively small.

2) *Training*: A set of 24 demonstrations were performed starting from different initial positions located on the horizontal plane of the table. The chess piece remained in a fixed location. Depending on the initial position, the chess piece was approached either downward or forward (as illustrated on Fig. 7). The set of demonstrations is represented in Fig. 3, right. The resulting velocity model is shown in Fig. 4, right. One can notice that there is no velocity feature that is common to all demonstrated trajectories. The acceleration model is shown in Fig. 5. This model captures well the fact that the vertical acceleration component depends on the position in the horizontal plane.

D. Results

Endowed with the system described above, the robot is able to successfully perform both tasks. For the first task, both the velocity and the acceleration models can produce adequate trajectories (see Fig. 6, left for examples). The system can adapt its trajectory online if the box is moved during movement execution (see Fig. 6, right). For the second task, examples of resulting trajectories are displayed in Fig. 7, right. In order to evaluate the generalization abilities of the systems, both tasks were executed from various different initial positions arbitrarily chosen on the horizontal plane of the table, and covering the space reachable by the robot. Fig. 8 shows the results and starting positions for both experiments. For the box experiment (left), the velocity model was successful for 22 out of the 24 starting locations (91%). The two unsuccessful trials,

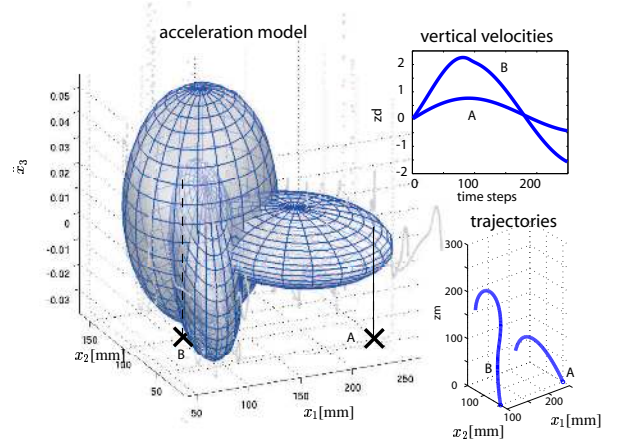


Fig. 5. In the center, the acceleration model for the second task. The ellipsoids show the Gaussian components at twice their standard deviation. Only three projections (out of nine) are shown. The vertical acceleration strongly depends on the position in the horizontal plane. On the lower right, two trajectories encoded by this model but starting from different positions A and B (indicated by the crosses) are shown. The corresponding vertical velocity profiles appear on the upper right. They differ significantly, as the model is not homogeneous across the horizontal plane.

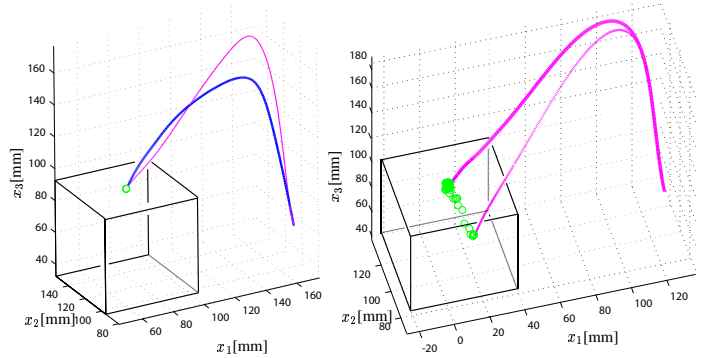


Fig. 6. Left: end-effector trajectories of the robot putting the object into the box. The thin line corresponds to the velocity model and the thick line corresponds to the acceleration model. Right: online trajectory adaptation to a target displacement using the velocity model. The circles indicate to location of the box, as tracked by the stereo-vision system. The thick line shows the produced trajectory and the thin line shows the original trajectory if the box remained unmoved. Similar results were obtained with the acceleration model.

indicated by empty circles, correspond to initial positions close to the work space boundaries. The acceleration model was successful for all trials (100%).

For the chess piece experiment (Fig. 8, right), the velocity model was successful for 5 out of 21 (24%) trials whereas the acceleration model was successful for 18 trials (86%). This performance gap is due to the fact that this task does not require a fixed velocity modulation. The adequate modulation depends on the position. This position-dependent modulation can be captured by the acceleration model, but not by the velocity model. As illustrated in Fig. 5, the acceleration model is able to produce different velocity profiles, depending on the starting position and is thus more versatile than the velocity model.

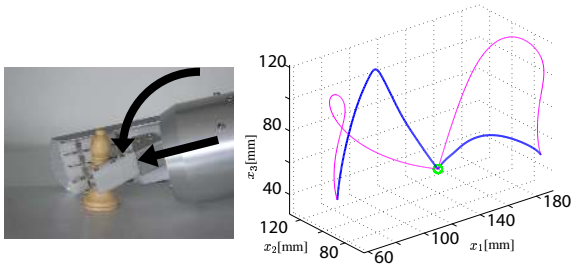


Fig. 7. Left: the chess piece to be grasped. For a successful grasp, the robot has to approach it as indicated by the arrows. Right: resulting trajectories for the grasping task, starting from two different initial positions. The acceleration model (thick lines) adapts the modulation to the initial position, while the velocity model (thin lines) starts upward in both cases. The trajectory produced by the velocity model and starting left of the target is not successful.

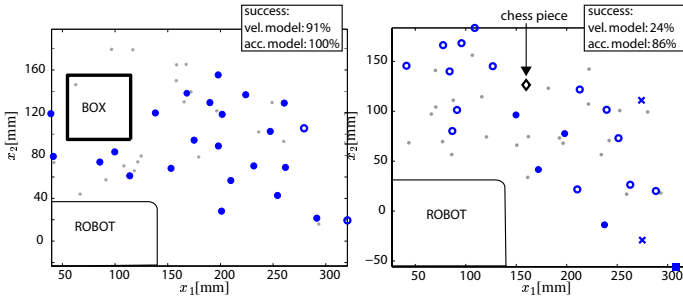


Fig. 8. The robustness to initial end-effector position for both tasks. The plots represent top views of the first (right) and second (left) experiment. The filled markers (circles or squares) indicate all initial positions for which the velocity model was successful. The circles (filled and non-filled) indicate all initial positions for which the acceleration model was successful. The crosses indicate initial end-effector position, for which both models failed. The dots indicate the starting positions of the training set.

V. DISCUSSION

Our results show that the framework suggested in this paper can enable a robot to learn constrained reaching tasks from kinesthetic demonstrations, and generalize them to different initial conditions. Using a dynamical system approach allows to deal with perturbations occurring during the task execution. This framework can be used with various task models and has been tested for two of them, the velocity model and the acceleration model. The results indicate that the velocity model is too simplistic if the task requires different velocity profiles when starting from different positions in the workspace. The acceleration model is more sophisticated and can model more constrained movements, but may fail to provide an adequate trajectory when brought away from the demonstrations in the phase space $(\mathbf{x}, \dot{\mathbf{x}})$. Other regressions techniques, such as LWR, could also be used. But if there are inconsistencies across demonstrations, simple averaging may fail to provide adequate solutions.

In its present form, the modulation factor between the dynamical system and the task model (γ) is not learned. Learning it from the demonstrations is likely to further improve the performance of the system, especially for tasks requiring a modulation at the end of the movement. It would also be desirable to have a system that extracts the relevant variables, and automatically selects the adequate model. A

first step in this direction has been taken in [16], where a balance between different sets of variables is achieved.

Of course, the adequacy of this framework is restricted to relatively simple tasks, such as those described in the experiments. More complicated tasks, such as obstacle avoidance in complex environments or stable grasping of particular objects require a detailed model of the environment and more elaborate planning techniques. The tasks considered for this framework are those that cannot be accomplished by simple point-to-point reaching, but still simple enough to avoid the complete knowledge of the environment. But this framework could be extended to learn more complicated tasks. In a first step in this direction, [17] investigates in simulation and on a simplified framework how Reinforcement Learning can deal with obstacle avoidance.

REFERENCES

- [1] A. Billard and R. Siegwart, Eds., *Robotics and Autonomous Systems, Special Issue: Robot Learning From Demonstration*. Elsevier, 2004, vol. 47, no. 2,3.
- [2] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002, pp. 1398–1403.
- [3] W. Ilg, G. Bakir, J. Mezger, and M. Giese, "On the representation, learning and transfer of spatio-temporal movement characteristics," *International Journal of Humanoid Robotics*, pp. 613–636, 2004.
- [4] A. Shon, J. Storz, and R. Rao, "Towards a real-time bayesian imitation system for a humanoid robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 2847–2852.
- [5] O. Jenkins, G. González, and M. Loper, "Tracking human motion and actions for interactive robots," in *Proceedings of the Conference on Human-Robot Interaction (HRI07)*, 2007, pp. 365–372.
- [6] D. Grimes, D. Rashid, and R. Rao, "Learning nonparametric models for probabilistic imitation," in *Advances in Neural Information Processing Systems (NIPS 06)*, 2006.
- [7] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, "Extraction of essential interactions through multiple observations of human demonstrations," *IEEE Trans. Ind. Electron.*, vol. 50, pp. 667–675, 2003.
- [8] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, no. 2,3, pp. 109–116, 2004.
- [9] C. Campbell, R. Peters, R. Bodenheimer, W. Bluethmann, E. Huber, and R. Ambrose, "Superpositioning of behaviors learned through teleoperation," *IEEE Transactions on Robotics*, 2006.
- [10] R. Burridge, A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *International Journal of Robotics Research*, 1999.
- [11] Z. Ghahramani and M. Jordan, "Supervised learning from incomplete data via an em approach," in *Advances in Neural Information Processing Systems 6*, J. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan Kaufmann Publishers, 1994.
- [12] A. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 370–384, 2006.
- [13] C. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 16, no. 1, pp. 93–101, 1986.
- [14] Y. Nakamura and H. Hanafusa, "Inverse kinematics solutions with singularity robustness for robot manipulator control," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 163–171, 1986.
- [15] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, 1978.
- [16] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 2, pp. 286–298, 2007.
- [17] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *RSJ Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.