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

This work was supported by the European Commission through the EU Integrated Projects ROBOT-CUB (FP6-IST-004370), Cogniron (FP6-IST-FET-002020) and ROBOT@CWE (FP6-034002).
**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]. This approach differs significantly from classical approaches for robot manipulation. Those approaches typically start by modeling the task, the relevant elements of the environment and the robot, as well as their dynamics. The problem is then to find the adequate robot command that will bring the whole system into a desired state specified by the programmer. This is usually done by using the plant model and sensor information to estimate the state of the world, and finding a control law specifying the command adequate to various states of the world. This law can be hard-coded, e.g. for juggling [2], grasping [3], 2D pushing and throwing [4], or obstacle avoidance for reaching [5]. But it can also be (partially) learned from (possibly simulated) exploration, e.g. for stable grasping [6] or object manipulation under wrench closure constraints [7]. In PbD, the idea is to try to extract an adequate control law from demonstrations of the task performed by a human. The demonstrations can indeed provide useful information, for example appropriate grasps in a grasping task [8] (see [1] for a further discussion of the use of PbD for robot control). 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 [9], HSTMM [10]), possibly in a latent space (GPLVM [11], ST-Isomap [12]) or some probabilistic model like HMMs [13] or Bayesian Networks [14]. 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 [15] or knowledge-based systems [16].

In our work, we position ourselves in between those two approaches. The tasks we consider (such as reach-to-grasp) require some interaction with the environment, while remaining relatively simple. Like the first approach, we train a motion model for the task, and like the second approach, we also use a hard-coded controller. We start with a basic built-in controller consisting in a dynamical system with a single stable attractor. We then learn a task model used to modulate the trajectories generated by the dynamical system in a way appropriate for a given task. 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 it is an improvement on [17], which also learns reach-to-grasp movements, but in a static setting.

The closest work to ours is [9], which uses a dynamical system for goal-directed reaching. There, a desired trajectory in joint space is obtained from a single demonstration and is embedded in a dynamical system, which can reproduce the qualitative features of this trajectory, while reaching a somewhat different target from a different initial position. In a previously published paper [18], we learned a velocity profile from demonstrations and used it to modulate a dynamical system acting on the end-effector. The novelty of the present contribution with respect to those last two papers is the following. First, while [9] learns a trajectory in joint space, and [18] is controlling in task space, here we propose a hybrid task and joint space controller, which can combine the advantages of both. The second and more fundamental difference lies in the level of generalization. Whereas [9] tries to reproduce a *single joint angle trajectory*, and [18] learns a *task specific velocity profile*, here we learn a whole *dynamical system* capturing the *correlations across multiple variables* for a given task. This enables us to present results that are not mere trajectory comparison (as in [9]), but that quantify the adaptivity of our controller at the level of task success rate.

We show experimentally that modeling the task as a dynamical system yields a more adaptive controller. In those experiments, the motions are demonstrated to the robot by a human user moving the robots’ limbs passively (kinesthetic training). We consider two tasks, 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 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 motion encoding either as a velocity profile or as an acceleration field. 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.

The above dynamical system is modulated by the variable $\dot{x}^m$ 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 weights 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 $\theta^g$, $\gamma$ has to tend to zero at the end of the movement. In the experiments described here, $\gamma$ is given by:

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

where $\gamma_0$ is the initial value of $\gamma$ and $\alpha_\gamma \in \mathbb{R}_{[0,1]}$ is a scalar. Since $\dot{x}^m$ lives in the end-effector space (and not in the joint space), the modulation is performed by solving the following constrained optimization problem.

$$\dot{\theta} = \arg\min_{\dot{\theta}} (1 - \gamma)(\dot{\theta} - \dot{\theta}^*)^T W_\theta (\dot{\theta} - \dot{\theta}^*) + \gamma (\dot{x} - \dot{x}^m)^T W_x (\dot{x} - \dot{x}^m) \quad \text{u.c.} \quad \dot{x} = J\dot{\theta},$$

where $J$ is the Jacobian of the robot arm kinematic function $K$ and $W_\theta \in \mathbb{R}^{n \times n}$ and $W_x \in \mathbb{R}^{m \times m}$ are diagonal matrices necessary to compensate for the different scale of the $x$ and $\theta$ variables. As a rough approximation, the diagonal elements of $W_x$ are set to one and those of $W_\theta$ are set to the average distance between the robot base and its end-effector.

The solution to this minimization problem is given by [20]:

$$\dot{\theta} = (W_\theta + J^T W_x J)^{-1} (W_\theta \dot{\theta}^* + J^T W_x \dot{x}^m)$$

where $W_\theta = (1 - \gamma) W_\theta$, $W_x = \gamma W_x$.

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

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

$$\ddot{x} = (W_\theta + J^T W_x J)^{-1} (W_\theta \ddot{\theta}^* + J^T W_x \ddot{x}^m)$$

where $\tau$ is the time integration constant (set to 1 in this paper). Since the position $x$ and velocity $\dot{x}$ depend on the acceleration $\ddot{x}$ at previous times, this representation introduces a feedback loop, which is not present in the representation given by (2).
where $W_x$ and $W_θ$ are given by (6) and (10), and $\hat{x}^m$ is given either by (2) (velocity model) or by (4) (acceleration model). Integration is performed using a first-order Newton approximation ($\hat{θ}^p = \hat{θ} + \tau \hat{θ}^p$).

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 [21] [22], 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 present 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 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.

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) [23], and the parameter values used were $\alpha_a = 0.06$, $α = 0.12$ and $β = 0.06$.

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
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 (and thus of the end-effector) lied on the 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.

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, 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
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, which models the task as a dynamical system rather than as a trajectory, is more sophisticated and can model more constrained movements. However, it may fail to provide an adequate trajectory when brought away from the demonstrations in the phase space \((x, \dot{x})\). 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 \((\gamma)\) 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 \([24]\), 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, \([18]\) investigates how Reinforcement Learning can deal with obstacle avoidance.

### References


