Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators

E. Gribovskaya, S.M. Khansari-Zadeh, Aude Billard

***Abstract*—Motion imitation requires reproduction of a dynamical signature of a movement, i.e. a robot should be able to encode and reproduce a particular path together with a specific velocity and/or an acceleration profile. Furthermore, a human provides only few demonstrations, that cannot cover all possible contexts in which the robot will need to reproduce the motion autonomously. Therefore, the encoding should be able to efficiently generalize knowledge by generating similar motions in unseen context.**

This work follows a recent trend in Programming by Demonstration in which the *dynamics* of the motion is learned. We present an algorithm to estimate multivariate robot motions through a Mixture of Gaussians.

The strengths of the proposed encoding are three-fold: i) it allows to generalize a motion to unseen context; ii) it provides fast on-line replanning of the motion in the face of spatio-temporal perturbations; iii) it may embed different types of dynamics, governed by different attractors.

The generality of the method to estimate arbitrary non-linear motion dynamics is demonstrated by accurately estimating a set of known non-linear dynamical systems. The platform-independency and real-time performance of the method are further validated to learn the non-linear motion dynamics of manipulation tasks with different robotic platforms. We provide an experimental comparison of our approach with an related state-of-the-art method.

***Index Terms*—Non-Linear Autonomous Dynamical Systems; Robot Programming by Demonstration; Learning by Imitation; Gaussian Mixture Model and Regression**

I. Introduction

The versatility of tasks that modern robots should accomplish has forced researchers to consider alternative methods for control: designing task-specific controllers becomes an inefficient and cumbersome solution. Therefore, preference gradually changes in favor of flexible and generic control methods that can adapt to various tasks and robots' geometries. If, in addition, the robot is expected to operate in the vicinity of or in collaboration with unskilled human users, control must be both intuitive and flexible to ensure safe and easy operability by the human.

Programming by Demonstration (PbD) has appeared as one way to respond to this growing need for intuitive control methods [Billard et al., 2008]. One of the requirements for the teaching methods in PbD to be effective is that the number of training examples should remain small (one considers between five and ten examples to be a bearable number for the trainer). Consequently, PbD either relies on prior knowledge to speed up learning, or results in a partial representation of the task which can be refined later.

PbD operates at different levels of the task representation: from copying low-level features of the motion [Sternad and Schaal, 1999, Ude et al., 2004, Calinon and Billard, 2008, Nguyen-Tuong et al., 2008, Schaal et al., 2003] to inferring the user's intention using a symbolic representation [Demiris and B.Khadhouri, 2006, Zollner et al., 2004]. In this paper, we focus on a low-level representation of motions, therefore, we further review work related to this direction of PbD. Low-level representations should determine the encoding of the demonstrated trajectories of motion so that they can be easily modulated to enable re-use of the skill in novel contexts. An overview of requirements for effective movement encoding has been summarized in [Ijspeert et al., 2001].

Most relevant to the present paper are the notions of *reusability* of the representation, i.e. the encoding should be easily transferrable to unseen context and the notion of *robustness* to perturbations, i.e. an ability of an encoding to ensure that a motion may be quickly adapted to perturbation and changes in a dynamic environment. The latter is a particularly important problem: perturbations can be caused by the physical inability of a robot to perform a pre-planned trajectory or due to the inherent uncertainties of the environment.

The term *perturbation* has been treated rather broadly in the current robotics research, however, to the best of our knowledge, no established classification of perturbations can be found in the literature. We therefore suggest classifying perturbations according to the following criteria. (1) *Spatial vs. temporal* perturbations, i.e. perturbations that either affect the position of the robot in space or modify the planned motion duration. These perturbations are often coupled, e.g. as the robot is pushed farther from the target, both spatial and temporal perturbations may occur. (2) *External vs. internal* (or self-generated), i.e. whether a perturbation has been applied externally (e.g. a robot has been pushed away while tracking a trajectory) or internally (e.g. if the motion planning algorithm autonomously generates a spatial perturbation to avoid an obstacle). (3) *Instantaneous vs. continuous*, i.e. whether the perturbation has an impulse character (e.g. in the case of a sudden push or a jerk) or the perturbation is applied continuously and thus systematically modifies the robot's motion (e.g. if a human applies a continuous force to slow down the robot's motion.) The suggested classification is not exhaustive; however, it may prove useful for a qualitative comparison of the existing motion planning methods.

Recent works on *feedback planning* [Tedrake et al., 2010, Brock et al., 2008] have emphasized the need to provide an encoding of robot motion that embeds the ability to adapt or even re-generate trajectories on the fly. While the planners are able to generate trajectories taking into account

different external and internal constraints, the planning might require significant computation time. This is an impediment for robotic applications that need the immediate response in the case of perturbations.

The approach of learning motions as dynamical systems that we follow here, was suggested as an alternative to classical planning algorithms [Schaal et al., 2007]. Autonomous dynamical systems used in Programming by Demonstration encode trajectories through a time-independent function that defines the temporal evolution of the motion. The advantages of the dynamical system motion representation as opposed to providing a robot with a single pre-planned trajectory are three-fold: (1) this representation exempts one from re-indexing trajectories in time while recovering from perturbation or during adaptation to new initial conditions (robustness to temporal perturbations). (2) Motion planning with dynamical systems allows for on-line adaptation to spatial perturbations, thus eliminating a need of additional algorithms to replan a complete trajectory or re-scale an existing one. (3) It offers means to generalize motions in areas of the workspace not covered during the demonstrations.

One of the main limitations on using dynamical systems for motion encoding is the fact that a learned representation can be unstable. A primary concern for motion generation with dynamical systems is, therefore, to ensure stability of an estimate. Once stability is ensured, dynamical systems are able to handle more complex constraints, like the presence of obstacles or robot's physical limitations (e.g. joint limits [Hersch and Billard, 2008]). Existing literature that derives a stable dynamical system does so by imposing an external stabilizer (e.g. the linear stabilizer of Dynamical Movements Primitives (DMP) [Ijspeert et al., 2001, Pastor et al., 2009]). A disadvantage of this approach is that the external stabilizer distorts a temporal pattern of a dynamics (see experiments in Section V-G). Our work focuses on the problem of building a stable dynamical system estimate of the motion that does not rely on an external stabilizer and, therefore, preserves a spatio-temporal pattern of a demonstrated motion[1].

The idea of the dynamical system motion representation has originally emerged in studies on human motion [Bizzi et al., 1984, Kelso, 1995, Todorov and Jordan, 2002]. These work suggest to view motion planning and execution as an intertwined problem, in which motions are generated by dynamical systems evolving in time and space. There is thus no explicit trajectory planning stage: a motion is generated on the fly according to the dynamical law.

The exact form of dynamical control in humans is still undeciphered. Early attempts at finding laws subserving human point-to-point movements developed computational models accounting for the "quasi-linear" trajectories, "bell-shaped" velocity profile[Bullock and Grossberg, 1988, Flash and Hogan, 1985] and $2/3$rd power law [Vivani and Terzuolo, 1982]. These however fell short at explaining reaching motions outside a 2D task space [Sternad and Schaal, 1999] and at accounting for the curvature of movements reaching for arbitrary points in space [Petreska and Billard, 2009]. Recent approaches take a less categorical view and no longer searching for a single invariant [Berret et al., 2008]. They come closer to robotics model, showing that kinematics and dynamics are tightly linked during control and that the particular law of motion underpinning control is task dependent [Admiraal and Kusters, 2004, Kang et al., 2005]

Following from the above, we implicitly ground our work on the assumption that human motions contain regularities that can be represented by a dynamical system. However, since there is no uniform approach to representing arbitrary via-point motions, we develop a method to learn a non-linear dynamical laws that encode kinematic invariants contained in motion data.

To model the natural variability of human motions, dynamical models often include a signal-dependent noise that is represented by a multiplicative Gaussian noise [Harris and Wolpert, 1998]. This signal-dependent noise, partly due to muscle fatigues and imprecision in sensor feedback, is considered as an inherent limitation of human motor control [Shadmehr et al., 2010]. Therefore, in our work, we assume that learning of the deterministic part that account for motion dynamics should be sufficient to design to corresponding robot control. Hence, we do not model the stochastic component responsible for a natural variability of motion.

In this paper, we consider the problem of estimating a *time-independent* model of motion through a set of first order non-linear multivariate dynamical systems. We exploit the strength of parametric statistical techniques to learn correlations across the variables of the system and show that this technique allows the determination of a coarse representation of the dynamics. We demonstrate advantages of such an approach as an alternative to the *time-dependent* methods, by ensuring robustness to external *spatio-temporal perturbations* through on-line adaptation of the motion. Here, under robustness to perturbations we particularly refer to the ability of the system to react to changes in the environment that are encapsulated by motion parameters, such as a desired target position and motion duration. Therefore, the system is able to cope with uncertainties in the position of a manipulated object, duration of motion, and perturbations associated with robot's body limitation (e.g., joint velocity and torque limits). According to the classification proposed earlier on, the proposed method aims at adapting to spatial and temporal perturbations which are externally-generated[2]; and applied instantaneously or continuously.

In our previous work Gribovskaya and Billard [2009], we investigated the application of the approach presented here to learn a control law for both position and orientation. The present manuscript presents the following important extensions to this work: (1) while in the conference paper we only outlined the iterative procedure, here we present a complete and in-depth description of the theoretical aspects of the algorithm and of the issue of stability which is at core of the approach and of its novelty;(2) we validate the approach

[1]Similar to other motion planning algorithms, our system requires an additional step to convert positions into motor commands by means of the inverse dynamics or a PID controller.

[2]Limited adaptation to internally-generated temporal perturbations is addressed through adaption to joint torque limits

of estimating theoretical dynamical systems and measure its robustness against several types of perturbations and noise; (3) we compare the method both theoretically and experimentally against related work; and (4) we apply the method in new robot experiments with the iCub robot.

This paper is divided as follows. Section II reviews related work on motion learning and estimation of dynamical systems. Section III-A starts with a formalization of the problem at hand and the particular approach of this work. This is followed by a technical description of the modeling approach: Section III-B introduces the learning approach to estimate the dynamics, while Section III-C presents an iterative algorithm to improve stability of the learned dynamics. Finally, in Section IV, we validate the method by estimating the motion dynamics from trajectories generated with given dynamical laws; in this way we may systematically verify approximation qualities of the method. We, further, show how the same framework can be used to learn the motion dynamics of manipulation tasks with different robotic platforms. To emphasize advantages of our approach as compared to the state-of-the-art methods in the field, we provide an experimental comparison with Dynamic Movements Primitives [Ijspeert et al., 2001, Pastor et al., 2009]. The legend used in graphs throughout the paper is summarized in Figure 1. The glossary is in Table I.

II. Related Work

To better delineate this paper's particular contribution to both machine learning and robotics, we focus our review on two major themes. First, to situate the dynamical systems approach taken in our work, we make a brief historical tour of the large volume of literature on modeling robot motion, contrasting time-dependent and time-independent representations. We then turn to the problem of estimating arbitrary dynamical systems and introduce the particular statistical technique used here. We briefly summarize the broad division across parametric and non-parametric statistical methods, and situate our choice of parametric method in this context.

A. Motion Learning

A core issue within robot control is ensuring that, if perturbed, the robot's motion can be rapidly and on-the-fly recomputed to ensure that the robot ultimately accomplishes the task at hand. Perturbations may lead the robot to either depart from its original trajectory (e.g. when slipping or hitting an object) or be delayed (e.g. when slowed down because of friction in the gears). In the rest of this paper, we will refer to the former type of perturbations as *spatial* perturbations and to the latter as *temporal* perturbations.

The vast majority of work on motion learning has addressed essentially the problem of being robust to spatial perturbation. Very little work has been yet done on handling temporal perturbations, which is core to the model we develop here. Next, we review these different approaches.

B. Time-dependent Modeling Approaches

Traditional means of encoding trajectories are based on spline decomposition after averaging across training trajectories [Hwang et al., 2003, Andersson, 1989, Yamane et al., 2004, Aleotti et al., 2005]. Spline decomposition remains a powerful tool for quick trajectory formation. It is, however, heavily dependent on a heuristic for segmenting and aligning the trajectories. Furthermore, spline representation, not being statistically-based, may have difficulties in coping with noise in data that is inherent in the robotic application.

Non-linear regression techniques were proposed as a statistical alternative to spline-based representation [Calinon et al., 2007, Schaal and Atkeson, 1998, 1994, Kulic et al., 2008]. These methods allow the systematical treatment of uncertainty by assuming the noise in data and, therefore, by estimating actual trajectories as a set of random variables with learned parameters.

However, similarly to spline-based approaches, most existing regression approaches to motion encoding consider as an input variable a time-index and virtually operate in "open-loop" (i.e. without a mechanism to adapt trajectories to perturbations or delays). The lack of the positional feedback makes these methods sensitive to both temporal and spatial perturbations. To compensate for this, one needs to introduce an external mechanism to track potential deviations from the desired trajectory during reproduction. Adaptation to deviations then relies on a heuristic to re-index the new trajectory in time or extrapolate in space. Such re-indexing or extrapolation often comes at the cost of deviating importantly from the desired velocity and acceleration profile, making the motion look "unnatural". Furthermore, finding a good heuristic is highly task-dependent and becomes particularly not-intuitive in multidimensional spaces [Schaal et al., 2003].

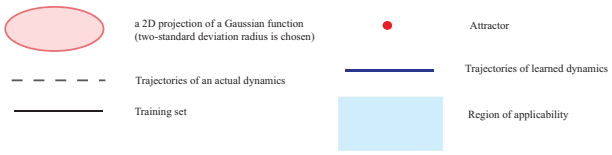

Fig. 1. The legend for the figures in the paper. Some figures are better seen in color, please, refer to the electronic version of the paper.

Time-independent models, such as autonomous dynamical systems (to which we will further refer to as DS), were recently advocated as an alternative to the above approaches[3]. Models based on DS are advantageous in that they do not depend on an explicit time-indexing and thus provide a closed-loop controller, while being able to model a broad class of non-linear behaviors. Removing the explicit time-dependency comes at a cost, as it re-introduced an old problem, namely the need to consider stability of the control policy.

Next, we review current approaches to DS modeling of robot motion and point out the limitations of these methods. For a detailed discussion on advantages and disadvantages of dynamical systems encoding of motion, see also [Ijspeert et al., 2001, Schaal et al., 2003, 2001, Schoner and Santos, 2001].

[3]DS formulation embeds the time-dependency of a system in the mathematical formulation of the problem by using time derivatives of the state variables.

C. Dynamical Systems Modeling of Motion

A number of recent approaches in PbD, including our prior work, investigate the use of dynamical systems for modeling robot motions [Ijspeert et al., 2001, Righetti et al., 2006, Dixon and Khosla, 2004, Ijspeert and Crespi, 2007, Hersch et al., 2008]. While [Dixon and Khosla, 2004] focuses on fitting the parameters of a first-order linear dynamical system into training data, the other above works tackle a problem of modulating a predefined linear dynamics with a non-linear estimate of a trajectory [Hersch et al., 2008] or a velocity profile [Ijspeert et al., 2001, Righetti et al., 2006]. The authors choose an uni-variate spring and damper system as an underlying linear dynamics. In such a way, they avoid an issue of stability of approximation that may occur if one learns an actual dynamics from data. However, this solution comes with its drawbacks: (1) uni-variate encoding discards information about correlation between degrees of freedom, that may be crucial for faithful reproduction (see Figure 30 for illustration of the uni-variate encoding problem). (2) Coupling of the output of a predefined linear DS with a regression estimate makes the overall system dependent on the temporal synchronization between the two signals and thus in effect time-dependent (see Table VI for a formal comparison between the proposed approach and the work [Ijspeert et al., 2001]). To handle temporal perturbations, one would need a heuristic to maintain the synchronization. This would, however, no longer guarantee that the overall system is globally asymptotically stable. (3) By ensuring that the stable DS takes precedence over the estimate when coming close to the attractor or after a given time period, one can show global stability of the complete estimate [Ijspeert et al., 2002]. In effect, the global dynamics of motion is increasingly dominated by the stable linear dynamical system, hence leading the motion to progressively depart from the learned dynamics. To ensure that the modulation still influences the dynamics of the motion when approaching the target, the method relies on using a large number of Gaussians spread across the data points.

In this paper, we develop an iterative procedure to learn a statistical estimate of an arbitrary multivariate autonomous dynamical system. We discuss the problem of stability of a learned estimate and propose an empirical procedure to verify stability and the region of applicability of the estimate. This relieves us from the need of using another a priori stable dynamical system and ensures robustness against spatial and temporal perturbations.

D. Estimating a Dynamical System

Data-driven methods for estimating dynamical systems consider multivariate input-output data as instances of a dynamical system and seek an estimate of the model that relates best these pairs of datapoints. Building a local approximation of the dynamics has been first reviewed within the time series analysis [Priestley, 1980, Chamroukhi et al., 2009, Ljung, 2004]. These works consider solely uni-dimensional data with a major motivation of predicting time series.

Analysis of dynamics has gradually shifted to state-space representation as it allows a representation of more sophisticated phenomena [Aoki, 1990, Crutchfield J.P., 1987]. The vast majority of these works focus on estimating *linear* dynamics [Dixon and Khosla, 2004, Ryoung K. Lim, 1998], a restrictive assumption for robotic applications. Recently, with the growing interest in chaos theory, more developed approaches have been proposed that allow approximation of complex dynamics [Crutchfield J.P., 1987, Wang et al., 2008]. While, several optimistic results in simulations have been presented [Carroll, 2007, Xie and Leung, 2005], their applicability to practical tasks with a small number of observed data containing noise remains to be verified.

The major body of numerical approaches of non-linear dynamical systems perform function approximation using different orthogonal polynomials (Chebyshev polynomials, B-splines [Lee, 1986], Radial Basis Functions [Buhmann, 2003] (RBFs)). Recently, many works have addressed the approximating properties of RBFs [Tomohisa et al., 2008, Travis et al., 2009, Wei and Amari, 2008]. RBFs have been proved to form universal approximators of any function on a compact set [Park and Sandberg, 1991]: any level of precision of the approximation may be achieved by considering an exhaustive number of basis functions; however, the quality of the approximation heavily depends on tuning a considerable amount of parameters. Thus, the problem of determining a tuning procedures optimum according to different criteria is a recurrent subject in the domain [Buhmann, 2003]. Furthermore, as the approximation with RBFs falls naturally into the category of non-parametrical methods discussed next, they suffer from the same types of limitations: RBFs better suits for approximation of uni-variate signals and quality of approximation rapidly deteriorates with an increase in the number of dimensions.

E. Statistical Encoding

Classically, the whole body of statical methods can be broadly divided into *parametric* and *non-parametric* approaches.

Non-parametric methods used in robot motion estimation include k-nearest neighbors [Moore, 1990], Gaussian Processes [Deisenroth et al., 2009, Nguyen-Tuong et al., 2008], Locally Weighted Regression, [Hardle, 1991, Muller, 1988, Schaal and Atkeson, 1998, 1994] and a combination of these [Nguyen-Tuong et al., 2008]. Non-parametric methods are advantageous over parametric methods as they make little assumptions about the form of the underlying distribution function to estimate. Moreover, due to the local nature of their estimate, non-parametric methods are well suited for accurate data fitting in low-dimensional spaces [Schaal and Atkeson, 1998, 1994]. Initially proposed for uni-dimensional problems, the above non-parametrical methods suffer from the curse of dimensionality [Bellman, 1957]: sparsity of training data in high-dimensional spaces makes accurate estimation of parameters almost impossible. Parametric methods, in contrast, are better suited to model a multivariate dataset. They, however, rely on heuristics to choose the underlying parameters efficiently.

The Gaussian Mixture Models (GMMs) and based on them Gaussian Mixture Regression (GMR) are parametric methods.

They are thus better suited for regression on multi-dimensional data [Sung, 2004]. Learning with GMM is classically done using Expectation-Maximization (EM), the iterative algorithm that optimizes the likelihood of the mixture of Gaussians over the data. Optimal performance relies, however, on choosing the number of Gaussians and on the stopping criterion of EM (see [McLahlan and Peel, 2000] for a review). While several methods have been proposed to automatically estimate these two parameters, with the Bayesian Information Criterion (BIC[4]) being the most generic, GMM estimation using EM may lead to suboptimal results and remain very sensitive to the initialization conditions. Here, we show that, for both our problem at hand and in practice, these known limitations are not an impediment and that an iterative method for choosing the number of Gaussians leads to good performance. Most importantly, we show that the method converges quickly and relies on very few parameters in comparison to parametric methods.

III. Method

A. Problem Statement

Consider that the state[5] of our robotic system can be unambiguously described by a variable ξ and that the workspace of the robot forms a sub-space X in $\mathbb{R}^N$.

Consider further that the state of our robotic system is governed by an *Autonomous Dynamical System* $\langle \mathcal{X}, f, T \rangle$ (as per Definition 1-2, Table I). Then, for all starting locations $\xi_0 \in X$, the temporal evolution of our robotic system is *uniquely determined* by the *state transition map* (Definition 2, Table I) $f(t, t_0, \xi_0) = \xi(t)$, $\forall \xi_0, \xi \in X$.

Let us further assume that the state transition map f is a non-linear continuous and continuously differentiable function and that the system is driven by a first order differential equation[6] with a single equilibrium point $\bar{\xi}$, such that:

$$\forall t \in T = [t_0; \infty]; \xi, \dot{\xi} \in X \subset \mathbb{R}^N \tag{1}$$

$$\dot{\xi}(t) = f(\xi(t)) \tag{2}$$

$$\dot{\bar{\xi}} = f(\bar{\xi}) = 0. \tag{3}$$

Let the set of M N-dimensional demonstrated datapoints $\{\xi^i, \dot{\xi}^i\}_{i=1}^M$ be instances of the above motion model corrupted by an multiplicative zero-means Gaussian noise. The problem consists then in reconstructing a noise-free estimate $\hat{f}$ of f based on the set of demonstrations. To this end, we will approximate the function in a subregion[7] $C \subset \Delta \subset X$, so that:

TABLE I
Glossary of definitions

Definition 1: The *state-space* $X \subset \mathbb{R}^N$ includes all possible instantiations of ξ, such that $\xi(t) \in X$ at each time step $t \in T = \mathbb{R}^+ = [0; \infty]$.

Definition 2: A *dynamical system* is the tuple $\langle \mathcal{X}, f, T \rangle$, with $f : t \to f^t$ a continuous map of X onto itself.

Definition 3: A dynamical system is *differentiable* if $\exists f : T \times X \to X$ such that for all $t_0 \in T, \xi_0 \in X$ the problem:

$$\dot{\xi} = f(t, \xi(t)), \;\; t \geq t_0, t \in T$$

$$\xi(t_0) = \xi_0$$

has a unique solution.

A dynamical system governed by a time-independent transition map with $f(t, \xi(t)) \triangleq f(\xi(t))$ is an *Autonomous Dynamical System*.

Definition 4. An *equilibrium state* $\bar{\xi} \in X$ of a dynamical system is such that

$$f(t, t_0, \bar{\xi}) = 0.$$

Definition 5. An equilibrium state $\bar{\xi} \in X$ is *stable* if $\exists \epsilon > 0$ and $\delta = \delta(\epsilon)$ such that

$$\forall \xi_0 \in B(\bar{\xi}, \delta) \Rightarrow f(\xi_0) \in B(\bar{\xi}, \epsilon),$$

$B(\bar{\xi}, \delta) \subset X$ is a hypersphere centered at $\bar{\xi}$ with radius δ. $\bar{\xi}$ is an *attractor* of f.

Definition 5. An *attractive state* is an equilibrium state $\bar{\xi}$ of a local flow, if there exists $\rho > 0$ such that:

$$\forall \xi_0 \in B(\bar{\xi}, \rho) \Rightarrow \lim_{t \to \infty} f(\xi_0) = \bar{\xi}.$$

$B(\bar{\xi}, \delta) \subset X$ is a hypersphere centered at $\bar{\xi}$ with radius δ. $\bar{\xi}$ is an *attractor* of f.

Definition 6. An equilibrium point $\bar{\xi}$ is *asymptotically stable* if it is both stable and attractive.

Definition 7. A set $\Delta \subset X$ is a *Region of Attraction (or Basin of Attraction)* of an equilibrium $\bar{\xi}$ if:

$$\Delta(\bar{\xi}) = \{\xi_0 \in X; \lim_{t \to \infty} f(\xi_0) = \bar{\xi}\}$$

See Figure 30-II for illustration.

Definition 8. A dynamical system is *globally asymptotically stable* at the equilibrium $\bar{\xi}$ if $\bar{\xi}$ is an asymptotically stable attractor and $\Delta \equiv \mathbb{R}^N$.

$$\hat{f} : C \to C \tag{4}$$

$$\hat{f}(\xi(t)) \approxeq f(\xi(t)), \forall \xi \in C.$$

C is further referred to as the *region of applicability* of a learned dynamics.

Without loss of generality, we can transfer the attractor to the origin[8], so that $\bar{\xi} = 0 \in C \subset X$ is now the equilibrium point of f and by extension of its estimate $\hat{f}$, i.e. $\hat{f}(0) = f(0) = 0$. If C is contained within the *region of attraction* Δ of $\bar{\xi}$ (see Definition 7, Table I), then the estimate $\hat{f}$ is asymptotically stable at $\bar{\xi}$ in C and any motion initiated from $\xi(t_0) \in C$ will asymptotically converge to the target $\bar{\xi}$.

[4]BIC introduces a penalty term for increasing the number of parameters in the model over the resulting improvement in the modeling performance.

[5]The state of a dynamical system represents the *minimum amount of information* required to describe the effect of past history on the future development of this system [Hinrichsen D., 2000].

[6]Considering solely *first order* dynamical systems is not restrictive to learning only first order relationships between trajectory and velocity, as one can always convert dynamics of an arbitrary order into a canonical system of first order ODEs.

[7]Estimating the dynamics in the whole state-space X would be practically infeasible due to the excessive number of demonstrations that this would require.

[8]To simplify the notation, we keep the same notation for the domains C and X after translation at the origin.

B. Approximating the Dynamics with Gaussian Mixture Regression

To construct $\hat{f}$ from the set of demonstrated trajectories, we follow a statistical approach and define $\hat{f}$ as a non-linear combination of a finite set of Gaussian kernels, using *Gaussian Mixture Models* (GMM).

GMMs define a joint probability distribution function $\mathcal{P}(\xi^i, \dot{\xi}^i)$ over training set of demonstrated trajectories $\{\xi^i, \dot{\xi}^i\}$, $i = 1..M$, M is the number of demonstrations, as a mixture of a finite set of K Gaussians $G^1..G^K$ (with μ^k and Σ^k being the mean value and covariance matrix of a Gaussian G^k):

$$\mathcal{P}(\xi^i, \dot{\xi}^i) = \frac{1}{K} \sum_{k=1}^{K} G^k(\xi^i, \dot{\xi}^i; \mu^k, \Sigma^k) \tag{5}$$

and

$$\mu^k = [\mu^k_\xi \; ; \; \mu^k_{\dot{\xi}}] \text{ and } \Sigma^k = \begin{pmatrix} \Sigma^k_\xi & \Sigma^k_{\xi\dot{\xi}} \\ \Sigma^k_{\dot{\xi}\xi} & \Sigma^k_{\dot{\xi}} \end{pmatrix} \tag{6}$$

Where each Gaussian probability distribution G^k is given by:

$$G^k(\xi^i_t, \dot{\xi}^i_t; \mu^k, \Sigma^k) = \frac{1}{\sqrt{(2\pi)^{2d}|\Sigma^k|}} e^{-\frac{1}{2}(([\xi^i_t, \dot{\xi}^i_t] - \mu^k)^T (\Sigma^k)^{-1} ([\xi^i_t, \dot{\xi}^i_t] - \mu^k))}. \tag{7}$$

The model is initialized using the *k-means* clustering algorithm starting from a uniform mesh and refined iteratively through Expectation-Maximization (EM) [Dempster et al., 1977]. To generate a new trajectory from learned GMMs, one can then sample from the probability distribution function given by Eq.5. This process is called Gaussian Mixture Regression (GMR).

EM estimation of GMM requires to inverse the covariance matrices, which is not possible when these matrices are singular. Such a singularity may result from severe data overfitting, e.g. when one of the Gaussian components collapses into a datapoint, in which case the log-likelihood function goes to infinity. The occurrence of covariance singularities depends on the training data and on the number of mixture components. In our work and particularly in the experiments reported in the paper, this problem did not arise due to 1) the nature of trajectory data, that were usually sampled at a high frequency and did not contain outliers; 2) a coarse resultant encoding characterized by a low number of mixture components.

Alternatively to EM training, one may employ a variational treatment of GMM [Attias, 1999] which assumes prior distributions over unknown parameters. Instead of estimating crisp values for covariances (the process that is prone to numerical instabilities), the variational approach estimates a family of possible covariance matrices.

Taking the posterior mean estimate of $\mathcal{P}(\dot{\xi}|\xi)$, the estimate of our function $\hat{\dot{\xi}} = \hat{f}(\xi)$ can then be expressed as a non-linear sum of linear dynamical systems, given by:

$$\hat{\dot{\xi}} = \sum_{k=1}^{K} h_k(\xi)(A_k\xi + B_k), \tag{9}$$

TABLE II
GAUSSIAN MIXTURE REGRESSION

Let us assume that we can for each input datapoint $\xi^{\mathcal{I}}$ match an output datapoint $\xi^{\mathcal{O}}$, the joint probability of input and output data is then modeled using Gaussian Mixtures. The probability that a datapoint $\eta = [\xi^{\mathcal{O}}; \xi^{\mathcal{I}}]$ belongs to the GMM is defined by

$$\mathcal{P}(\eta) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\eta; \mu_k, \Sigma_k) =$$

$$= \sum_{k=1}^{K} \pi_k \, \frac{1}{\sqrt{(2\pi)^D |\Sigma_k|}} \, e^{-\frac{1}{2}\left((\eta-\mu_k)^\top \Sigma_k^{-1} (\eta-\mu_k)\right)}$$

where π_k are prior probabilities and $\mathcal{N}(\mu_k, \Sigma_k)$ are Gaussian distributions defined by centers μ_k and covariance matrices Σ_k, where input and outputs components are represented separately as

$$\mu_k = \begin{bmatrix} \mu_k^{\mathcal{I}} \\ \mu_k^{\mathcal{O}} \end{bmatrix}, \quad \Sigma_k = \begin{bmatrix} \Sigma_k^{\mathcal{I}} & \Sigma_k^{\mathcal{I}\mathcal{O}} \\ \Sigma_k^{\mathcal{O}\mathcal{I}} & \Sigma_k^{\mathcal{O}} \end{bmatrix}.$$

Gaussian Mixture Regression allows to compute for a given input variable $\xi^{\mathcal{I}}$ and a given component k, the expected distribution of $\xi^{\mathcal{O}}$ as:

$\mathcal{P}(\xi^{\mathcal{O}}|\xi^{\mathcal{I}}, k) \sim \mathcal{N}(\hat{\eta}_k, \hat{\Sigma}_k)$, where

$$\hat{\eta}_k = \mu_k^{\mathcal{O}} + \Sigma_k^{\mathcal{O}\mathcal{I}} (\Sigma_k^{\mathcal{I}})^{-1} (\xi^{\mathcal{I}} - \mu_k^{\mathcal{I}}),$$
$$\hat{\Sigma}_k = \Sigma_k^{\mathcal{O}} - \Sigma_k^{\mathcal{O}\mathcal{I}} (\Sigma_k^{\mathcal{I}})^{-1} \Sigma_k^{\mathcal{I}\mathcal{O}}.$$

where $h_k = \mathcal{P}(k|\xi^{\mathcal{I}})$ is the probability of the component k to be responsible for $\xi^{\mathcal{I}}$

$$h_k = \frac{\mathcal{P}(k)\mathcal{P}(\xi^{\mathcal{I}}|k)}{\sum_{i=1}^{K} \mathcal{P}(i)\mathcal{P}(\xi^{\mathcal{I}}|i)} = \frac{\pi_k \, \mathcal{N}(\xi^{\mathcal{I}}; \mu_k^{\mathcal{I}}, \Sigma_k^{\mathcal{I}})}{\sum_{i=1}^{K} \pi_i \, \mathcal{N}(\xi^{\mathcal{I}}; \mu_i^{\mathcal{I}}, \Sigma_i^{\mathcal{I}})}.$$

Alternatively, by using the linear transformation property of Gaussian distributions, the conditional expectation of $\xi^{\mathcal{O}}$ given $\xi^{\mathcal{I}}$ can be defined approximately defined by a single normal distribution with the parameters:

$$\hat{\mu} = \sum_{k=1}^{K} h_k \, \hat{\mu}_k \, , \quad \hat{\Sigma} = \sum_{k=1}^{K} h_k^2 \, \hat{\Sigma}_k. \tag{8}$$

where $A_k = \Sigma^k_{\dot{\xi}\xi}(\Sigma^k_\xi)^{-1}$, $B_k = \mu^k_{\dot{\xi}} - A_k\mu^k_\xi$, $h_k(\xi) = \frac{\mathcal{P}(\xi; \mu^k_\xi, \Sigma^k_\xi)}{\sum_{k=1}^{K} \mathcal{P}(\xi; \mu^k_\xi, \Sigma^k_\xi)}$, $h_k(\xi) > 0$, and $\sum_{k=1}^{K} h_k(\xi) = 1$.

Such a rewriting will prove useful when studying the stability of the estimate, as will be discussed in Section III-C.

A geometric illustration of the GMR inference in the case of single Gaussian is presented in Figure 2 and the GMR procedure is summarized in Table II. Figure 3 further illustrates the encoding process from GMM to GMR for a non-linear dynamical system with a single attractor.

C. Stability Analysis

Stability analysis of *linear* dynamical systems is well-studied subject [Khalil, 1996]: one either constructs a Lyapunov function for the system or analyzes the eigenvalues of the control matrix.

In contrast, there is no unique method to analyze the stability of *non-linear* dynamical systems and theoretical solutions exist only for particular cases. Classically, stability analysis of non-linear dynamical systems is performed in the two steps: (1) the system is linearized in a neighborhood around the points of interest (the attractors) and asymptotic stability of these attractors is verified; second, analysis of the region around the attractors is done to determine the region of attraction of the actual non-linear system.

Methods to analytically estimate the regions of attraction (see Definition 7, Table I) are often based on the construction of a Lyapunov function gradually expanding its region of

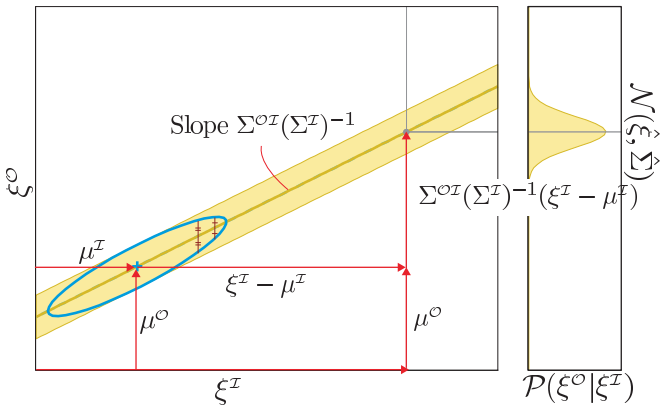

Fig. 2. The geometric illustration of Gaussian Mixture Regression inference (see also Table II). GMR approximates our dynamical systems through a non-linear weighted sum of local linear models: each regression matrix $A_k = \Sigma_k^{\mathcal{OI}}(\Sigma_k^{\mathcal{I}})^{-1}$ defines coefficients of the local linear fit. Here, we display the effect of fitting with a single Gaussian a pair of input and output signals $\xi_i^O \in R^M$, $\xi_i^I \in R^P$ respectively. The projection of the regression signal to the subspace spanned by $\{\xi_i^{O,m}, \xi_i^{I,p}\}$ is a line with a slope given by the elements A_k^{mp} of the regression matrix (i.e., $\xi_i^{O,m} = A_k^{mp}\xi_i^{I,p}$). The mixture of covariance matrix in GMM defines a probabilistic envelope around the regression signal. Thus, to each input ξ_i^I is associated a probability distribution function for output $\mathcal{P}(\xi_i^O|\xi_i^I)$, with mean ξ_i^O. In the present work, we exploit the envelope to determine the boundaries for our generalized inverse kinematics solution when the solution is not exactly the regression signal ξ_i^O.

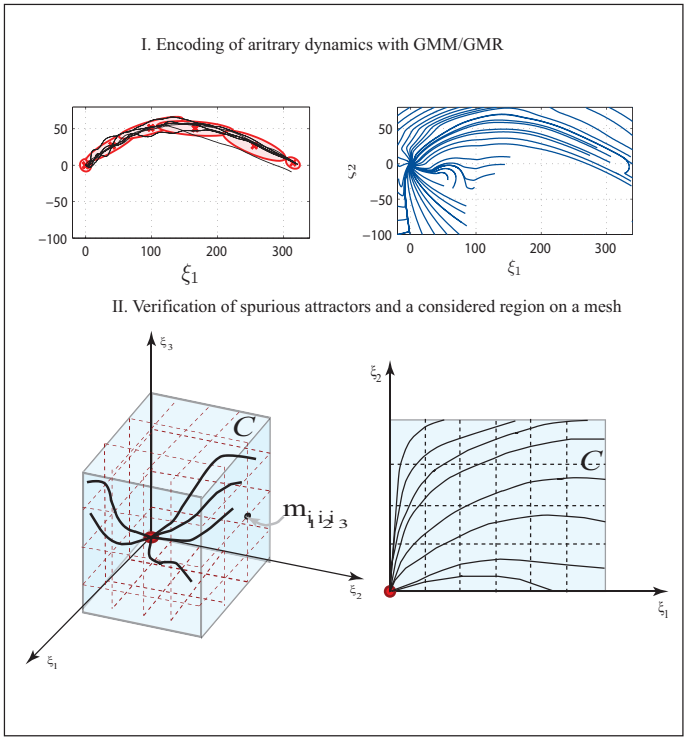

Fig. 3. I. Illustration of a GMM/GMR encoding of an arbitrary dynamics. *Top left*: Two-dimensional projection of the data with superimposed the Gaussian Mixture envelope. Top right: All trajectories regenerated using Gaussian mixture regression when starting from 20 different locations in space converge correctly to the the origin, the attractor of the system. *Bottom left and right*: in blue (light grey in a black-and-white version), the region of applicability C that embeds all demonstrated trajectories. To empirically determine if C is a region of attraction, C is sampled equally and one measures if all trajectories originating from each of sampled point converges correctly to the target.

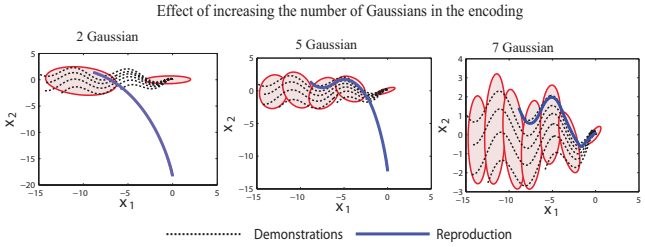

Fig. 4. Improvement in the stability of approximation with the increase in the number of Gaussian components

validity [Bai et al., 2007, Giesl, 2008, Genesio et al., 1985]. Such a procedure however produces a rather coarse estimation of the region of attraction and may fail to identify regions with non-convex boundaries. Alternative approaches take a geometrical perspective by reversing the flow of motion (by analyzing a dynamical function with a opposite sign) starting from the attractor and finding repellers and boundaries for a region of attraction from the reversed trajectories [Loccufier and Noldus, 2000]. These methods are more accurate but require considerable computation time, a known structure of an attractor's landscape (number of existing attractors and repellers).

Theoretical estimation of the region of attraction in the general case of multivariate non-linear systems is thus still an open problem. In practice, one relies on numerical procedures for evaluating whether a given region of applicability is a region of attraction. Here, we follow such an approach.

We start from the observation that GMR gives us a non-linear weighted sum of linear dynamical systems; see Eq. 9. Stability of the system is governed by the GMR parameters (the matrices A_k, B_k and mixing coefficients h_k), which are learned during training. Since the stability of the learned dynamics depends on the parameters of the training algorithm (Expectation Maximization) in Section III-D) we will show that a modification of the GMM procedure to build the mixture results in an estimate locally stable around the target.

1) Local stability at the origin: Following from the hypothesis that the origin is an attractor of the true control law $\dot{\xi} = f(\xi(t))$, we must ensure that its estimate given by (9) is also stable at the origin. Recall that for a point to be an attractor of the system (see Definition 5, Table I), there must exist a region around it where all trajectories are asymptotically stable.

Let us assume that in the neighborhood of the origin the system is governed solely by the last Kth gaussian [9]. In other words, let us assume there exists a neighborhood of the origin, where for points ξ in this neighborhood all mixing coefficients

[9]In practice, as we seek to avoid the over-fitting, the Gaussians are sufficiently set apart, therefore at the origin the influence of all other Gaussians except for the last one becomes numerically zero.

expect the Kth are zeros: $\exists B(\epsilon)$such that $\forall \xi \in B(\epsilon)\ h_k(\xi) \simeq 0\ k = 1..K-1$, where $B(\epsilon)$ is a hypersphere of radius ϵ. In this region, the system governed by Eq.9 reduces then to:

$$\dot{\xi} = A\xi + B \tag{10}$$

with $A = \Sigma_{K,\dot{\xi}\xi}\Sigma_{K,\xi}^{-1}$ and $B = \mu_{K,\dot{\xi}} - A\mu_{K,\xi}$.

The system above, driven by Eq. 10, will be asymptotically stable if the eigenvalues of the symmetric matrix $\tilde{A} = (A + A^T)/2$ are all strictly negative. For a $m \times m$-dimensional matrix to be negative definite, all its i-th order leading principal minors should be negative if i is odd and positive if i is even; stability, therefore, is guaranteed when the following set of constraints is satisfied:

$$\|\tilde{A}_{[1:i,1:i]}\|(-1)^i < 0 \quad \forall i = 1, ..., m \text{ that is satisfied if } (11)$$
$$(1)\ \tilde{a}_{ii} < 0 \text{ and } (2)\ \tilde{a}_{ij} \ll \tilde{a}_{ii}\ \forall\ i, j = 1, ..., m \text{ and } i \neq j, \tag{12}$$

where $\tilde{A} = \{\tilde{a}_{ij}\}_{i,j=1}^{N}$.

Figure 5 illustrates geometrically the effect of the local stability condition on the dynamics of motion and the form of the Gaussian. When projected on the $\{\xi_i, \dot{\xi}_i\}$ axes, the Gaussian corresponds to an ellipse with the main axis forming a negative slope. This results in a homogenous flow of motion toward the attractor along all dimensions.

For EM to result in such an elongated Gaussian, training data must homogeneously cover the space of motion around the target. This means that one should show the robot how to approach the target by uniformly starting all around the target. In practice, because the training set is finite and gives only a partial coverage of the state space, GMM estimate will be imprecise, resulting in both a shift of the slope of the Gaussian and a shift of the attractor's location, see Figure 5. Additional measures should, thus, be taken to guarantee the convergence to the target, which we describe next.

D. Practical approach to ensuring and analyzing stability

1) Ensuring local stability empirically: To overcome the lack of uniformly distributed training data around the origin in the experiments presented here, we generate additional so-called *synthetic* data by rotating a subset of training data, selected within a small neighborhood, around the origin. In addition, we set the center of the last Gaussian of the GMM at the target, i.e. at the origin ($\mu_{K,\xi} = \mu_{K,\dot{\xi}} = 0$), and do not update this center during training. This procedure is illustrated in Figure 5.

The system driven by the truncated dynamics is given by Eq.(10) and a system generated through this procedure is ensured to be asymptotically stable within a neighborhood around the origin. Next, we describe a procedure by which we can empirically estimate boundaries of the region of applicability C.

2) Determining the region of stability empirically: As mentioned in Section III-A, estimating dynamics in the whole state-space X is impractical. Instead, we will estimate stability locally within a subset $C \subset X$. C includes training data points and lies inside the robot's workspace. Initialization of C is data-driven: size of C along each dimension is defined by the amplitude of the training dataset along this dimension.

TABLE III
MODEL TRAINING

1	Collect a dataset of demonstrations and initialize C (see Section III-D2).
2	Add synthetic data around the target
3	Choose an initial number of GMM components K ($K = 2$ in the experiments reported here)
4	**LOOP** until stable approximation is found
5	Train the joint probability $\mathcal{P}(\xi, \dot{\xi})$ with Expectation Maximization [Dempster et al., 1977]:
6	Verify local stability at the origin Eq. (12)
7	**IF** (the origin is not asymptotically stable) **THEN** increase the number of GMM components $K = K + 1$
8	**ELSEIF** (estimate of C does not include all training trajectories) **OR** ($\exists$ spurious attractors inside the region C) **THEN** add training data **AND** retrain
9	**END**
10	**END**

After training, the initial guess regarding C should be restimated, to empirically verify that C is a region of attraction of the origin and that it does not include any other attractors . We follow a numerical procedure in which we integrate trajectories forward starting from a uniform mesh defined on the boundaries, and verify that all the trajectories converge toward the origin.

To do this, we construct a mesh M covering boundaries of C: $M(\tau_1..\tau_N) = \{(\xi_{i_1}^1..\xi_{i_N}^N) = (i_1\tau_1..i_N\tau_N),\ i_1 = 1..n_1, ...,\ i_N = 1..n_N\}$, where $\tau_1 = c_1/n_1\ ..\tau_N = c_N/n_N$, $c_1\ ..\ c_N$ – size of each of dimensions of C; $n_1..\ n_N$ – size of the mesh along each of dimensions in $\mathbb{R}^N$ (see Figure 3-II).

We integrate trajectories starting from each node $(\xi_{i_1}^1..\xi_{i_N}^N)$ on the mesh M and verify that the velocity is zero only at the origin, thus ensuring that only the origin of the system is an attractor. If this condition is satisfied all trajectories starting inside C will not leave the boundaries, due to the properties of differential equations.

To improve stability, we increment the number of Gaussians K and re-estimate the system using EM. Augmenting the number Gaussians allows a more precise encoding of the dynamics locally along the trajectory; see Figure 4. Since instabilities result often in the motion exiting the desired trajectory (e.g. if there are sharp turns in the trajectory that have been poorly approximated by the mixture), increasing the granularity of the encoding ensures that the system will be better guided along the various non-linearities of the trajectory.

Table III summarizes the steps of the complete procedure by which we iteratively test and re-estimate the system to improve and ensure local stability within the domain C.

IV. EXPERIMENTAL RESULTS

To validate the performance of the proposed method itself without blurring it with noise inherent to human demonstrations, we first tested its ability to reconstruct given theoretical

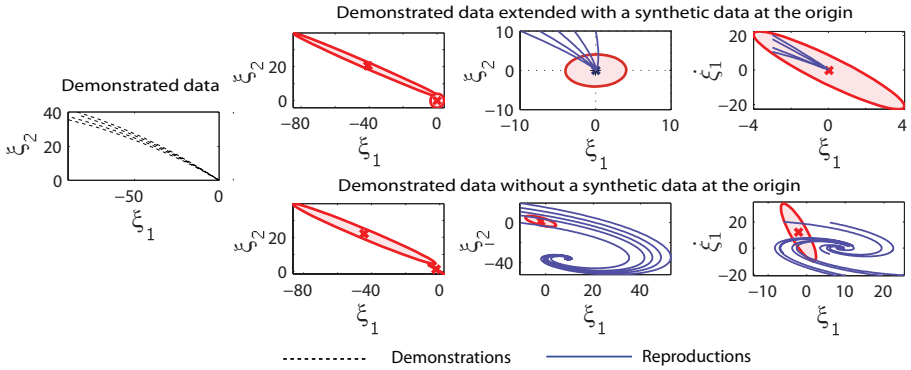

Fig. 5. Influence of the accurate positioning of the last Gaussian at the origin. *Top*: the last gaussian is positioned at the origin through addition of synthetic datapoints, that guarantees asymptotic stability of the system in the neighborhood of the origin, as can be seen from the vector field trajectories (the very right graph). *Bottom*: however, the real data asymptotically converge to the origin (the very left graph), the statistical EM does not automatically position the last Gaussian at the origin, that leads to the convergence to the spurious attractor (the very right graph).

dynamical systems. With a known system we may generate a clean training set, learn an approximation of the dynamics and further compare how well the learned dynamics approximate the real one.

Further, we verify the applicability of the method to robotics by teaching two robots manipulation tasks. We report on each of these next.

A. Learning Theoretical Dynamics

The method was validated to estimate four two-dimensional dynamical systems (*Systems 1-4*) and one three-dimensional dynamical system (*System 5*), each of them contains different number of attractors and exhibits different stability properties. In each case, we generated six trajectories using the theoretical dynamics and used these for training the GMM. When the dynamical system had more than one asymptotically stable attractor, trajectories were generated only in the subpart of the state space around one of them.

Note, the legend for Figures 6 - 9 is described in Figure 1. Each of the figures encompasses, in the first row, plots giving a general view of the original dynamics with vector fields (a) and three-dimensional phase plots (b-c), in the second row, a view of the GMM superimposed to the training data, and in the 3rd row, vector field (a) and phase plots (b-g) of the the estimated dynamics superimposed on the original dynamics.

System 1.

$$\begin{aligned}\dot{x}_1 &= -x_1 + 2x_1^2 x_2; \\ \dot{x}_2 &= -x_2.\end{aligned} \tag{13}$$

The system has a single locally asymptotically stable equilibrium point at the origin. We approximate the dynamics of this system in a region $[-4; 0] \times [0; 2]$, where it is locally asymptotically stable. Results are presented in the Figure 6.

System 2

$$\begin{aligned}\dot{x}_1 &= 700 - 2x_1 + 200x_2 e^{\frac{25x_1 - 10^4}{x_1}}; \\ \dot{x}_2 &= 1 - x_2 - x_2 e^{\frac{25x_1 - 10^4}{x_1}};\end{aligned} \tag{14}$$

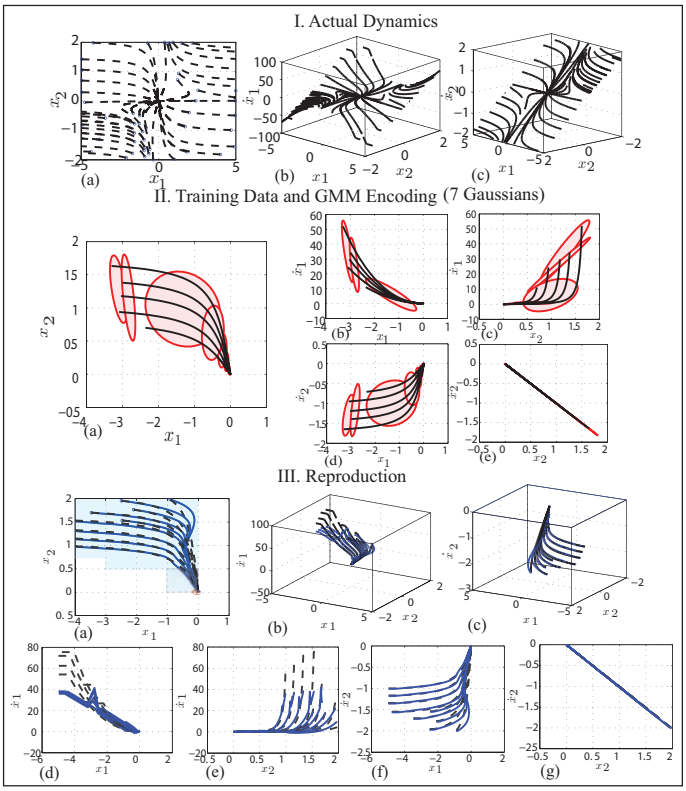

Fig. 6. *System 1.* The proposed method encodes this system with 7 Gaussians; the learned system exhibits good precision in the area covered by demonstrations, outside this area the precision is also admissible except for a region in the direct proximity to y-axis, where actual trajectories represent an excess curvature as approaching to the equilibrium, e.g., a trajectory starting at the bound $x_2 = 2$. In this region, a flat part of trajectories is reproduced well, though the steep parts that were not demonstrated are attracted towards the region covered by the training set.

The system has two equilibrium points – one asymptotically stable ($x_1 = 335$; $x_2 = 0.089$) and one unstable ($x_1 = 489$; $x_2 = 0.5$). We approximate the dynamics in the region $[0; 400] \times [-2; 2]$, where it is locally asymptotically stable. Results are presented in Figure 7.

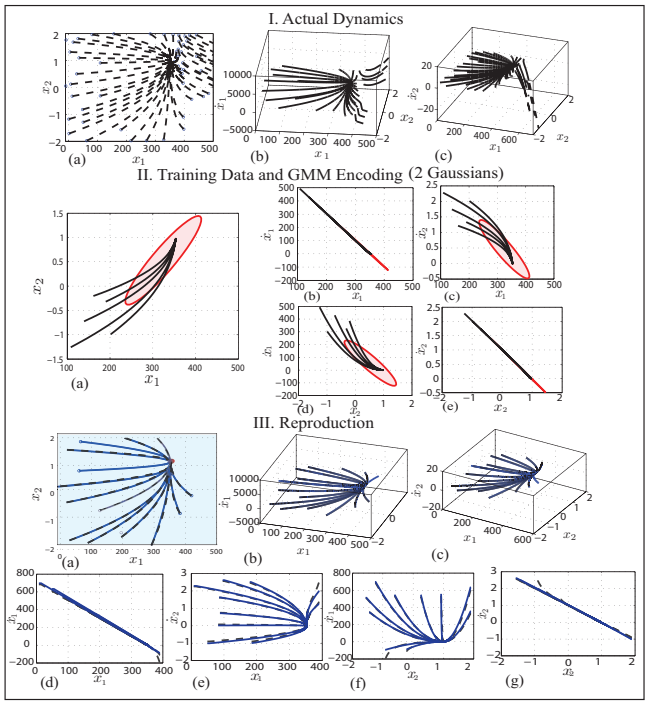

Fig. 7. *System 2*. As the behavior of the system in the considered area is relatively simple, 2 Gaussians are sufficient to achieve the good performance, even in areas unseen during demonstration. Interestingly, the learned dynamics is extrapolated very well beyond the area covered by the training set.

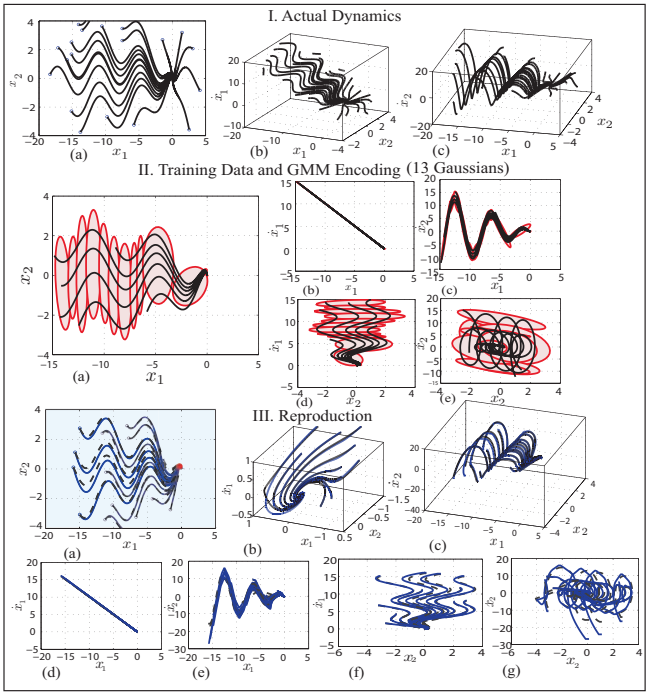

Fig. 9. *System 4*. The system is strongly non-linear, 13 Gaussians are necessary to achieve a good precision in the considered region. Complex dynamics and increased number of Gaussians lead to less strong generalization abilities of the method. Indeed, trajectories started beyond the region covered by the training set tend to depart from the real trajectories generated by the dynamics, it is particularly noticeable in the velocity space, see section III-(g). However, even in this non-trivial case the system generate admissibly good results from few demonstrations.

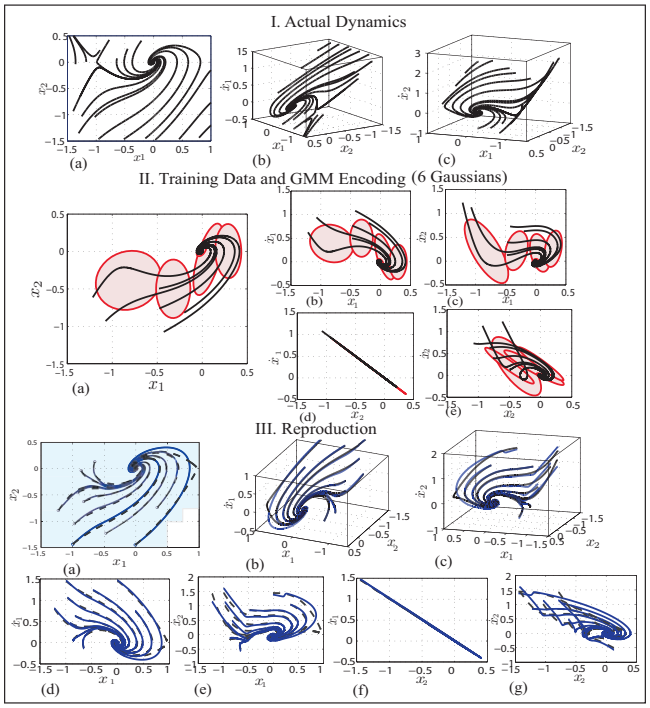

Fig. 8. *System 3*. Despite manifest non-linearity in the trajectories, the dynamics is successfully approximated with 6 Gaussians. Note, even unseen, circular shape trajectories (starting around $x_2 \approx 0$) are reproduced correctly in both position and velocities spaces.

System 3

$$\begin{aligned} \dot{x}_1 &= -x_2; \\ \dot{x}_2 &= x_1 - x_1^3 - x_2; \end{aligned} \tag{15}$$

The system has three equilibrium points - two unstable ($x_1 = -1; x_2 = 0$ and $x_1 = 1; x_2 = 0$) and one asymptotically stable $x_1 = 0; x_2 = 0$. We approximate the dynamics of this system in a region $[-1.5; 1] \times [-1.5; 0.5]$, where it is locally asymptotically stable. Results are presented in Figure 8.

System 4

$$\begin{aligned} \dot{x}_1 &= -x_1; \\ \dot{x}_2 &= -x_1 \cos x_1 - x_2; \end{aligned} \tag{16}$$

The system exhibits strong nonlinearity due to the cosine term; the system is globally asymptotically stable and converges asymptotically to the origin. We approximate the dynamics of this system in a region $[-20; 0] \times [-4; 4]$. Results are presented in Figure 9.

System 5

$$\begin{aligned} \dot{x}_1 &= -x_1 - x_2 + x_3^2; \\ \dot{x}_2 &= x_1 + 10 \cos x_2 * x_2 - x_3^2; \\ \dot{x}_3 &= x_1 + 2x_2 - x_3; \end{aligned} \tag{17}$$

Locally asymptotically stable three-dimensional dynamics with a single attractor at $[12.98; -7.75; -2.5213]$. We approximate the dynamics of this system in a region $[-20; 30] \times$

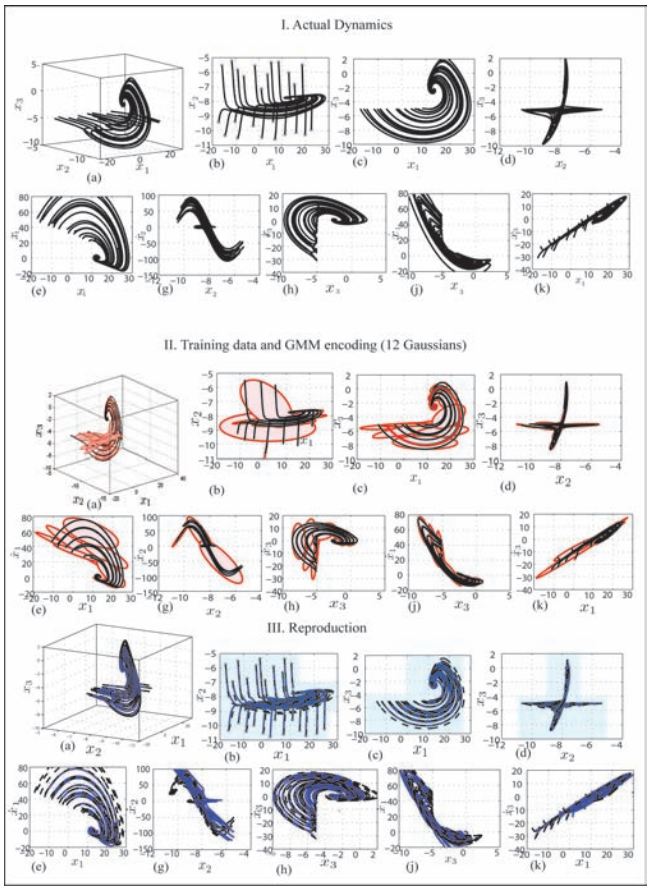

Fig. 10. *System 5*. Strongly non-linear 3D dynamics. In this case, a slight increase in a number of demonstrations allows for accurate approximation and generalization.

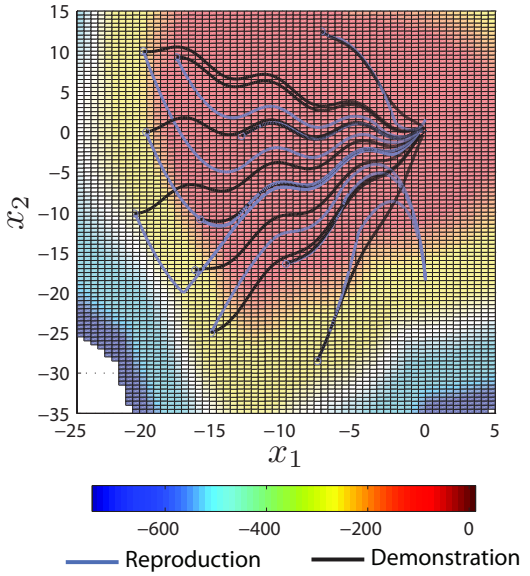

Fig. 11. Extrapolation properties of the GMMs encoding (better see in color). A color map reflects changes in values of the the likelihood (18) of datapoints, the dark-red (dark grey in the center) area represents an area of the most reliable inference regarding the velocity. For reconstructed trajectories starting outside this area, the deviation from the actual dynamics may be considerable. Interestingly, in the region of attraction of the origin, trajectories are strongly attracted towards a region covered by the training set. It is a useful property for practical applications as this allows to predict the behavior of the system outside the region covered during training, hence reducing the amount of training data required.

$[-11; -5] \times [-10; 2]$. Results of the learning process are presented in the Figure 10.

1) Quantification and discussion of results: Quantification of results achieved on both theoretical systems and actual robotic motions are presented in Table IV. As it can be seen all systems result in a coarse representation of motion dynamics through a relatively small number of Gaussians. Moreover such a sparse representation achieves good precision when reproducing the actual dynamics for both positional and velocity profiles.

As shown in Figures 6-10, the system can generalize outside the training domain (inside the stability domain and the domain of reliable inference as discussed below). This property is particularly useful for practical applications as this allows to predict the behavior of the system outside the region covered during training, hence reducing the amount of training data required. In the examples covered here, only 6 training trajectories were required in each case.

Note that, since the dynamics is learned from data covering only a subpart of the domain, it does not necessarily have the same attractor landscape and the region of attraction across the complete domain as the original system, even if it accurately approximates the original system locally. For example, in System 3, the original dynamical system has three equilibrium points, while its approximation has a unique asymptotically stable equilibrium. To overcome this, one may provide additional demonstrations covering dynamics in the neighborhood of the other equilibriums: Figure 15 presents results of learning the dynamics around the two different attractors of *System 5*. The demonstrations were provided in the neighborhood of the two asymptotically stable attractors; during learning, positions of two Gaussians were fixed on the attractors, and the algorithm was running to verify local asymptotical stability of both attractors. The regions of approximation C were analyzed separately for each attractors. The learned system managed to accurately grasp the complex dynamics, further, it allowed to separate the two flows of trajectories leading to different attractors based on the initial conditions of motion.

In addition to stability of reproduction, one should keep in mind that the considered region of applicability should not exceed a region where the likelihood of observing new data allows performing a confident inference regarding the velocity. In Figure 11 we depict how the likelihood changes beyond the region covered by the training set. Likelihood was computed as follows:

$$\mathbf{L}(\xi) = \log\left[\max_i h_i(\xi)\right]. \tag{18}$$

L gives a measure of the maximum probability of a point ξ to belong to any of the K Gaussians. The region where L exceeds a given threshold[10] represents the region where the system can still make a confident probabilistic inference. Note that all the trajectories that start in areas where L is too small

[10]We took an empirically chosen threshold of -10.

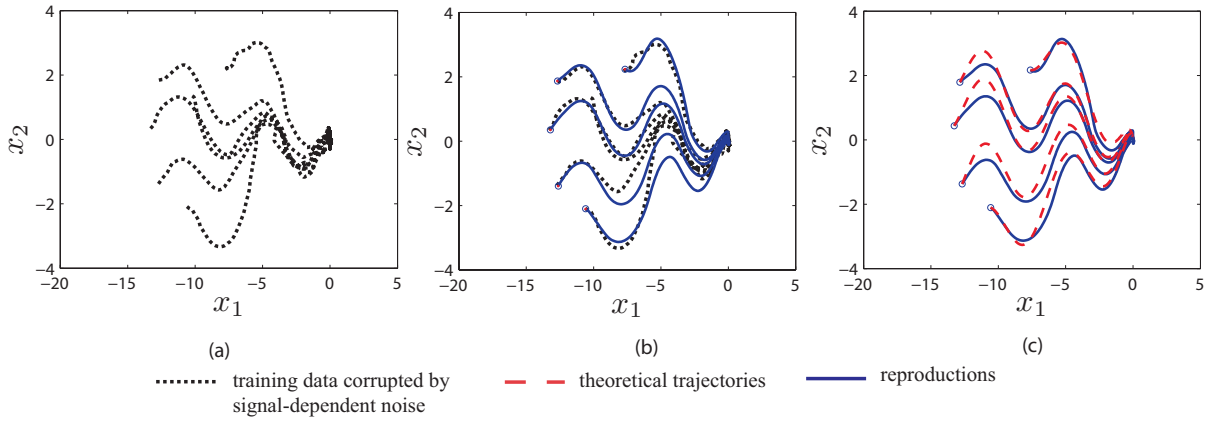

Fig. 14. (a) Training trajectories of *System 4* are corrupted with a signal-dependent noise with variance $\sigma^2 = 20$. (b) The trajectories generated by a learned dynamics are superimposed with the training trajectories. (c) The trajectories generated by the learned dynamics are superimposed with the theoretical trajectories. Note, that the proposed method manages to accurately extract a typical pattern of the motion.

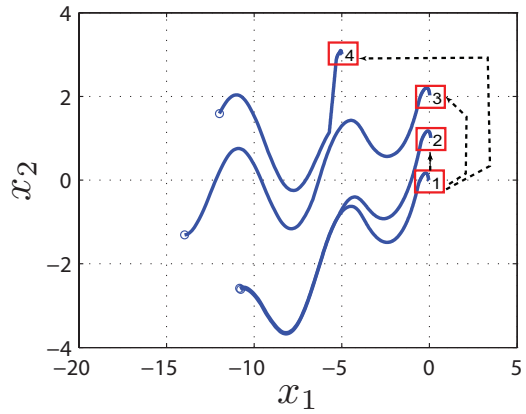

Fig. 12. Robustness to perturbations. The target is shifted several times (to positions 2, 3, 4) after the onset of motion.

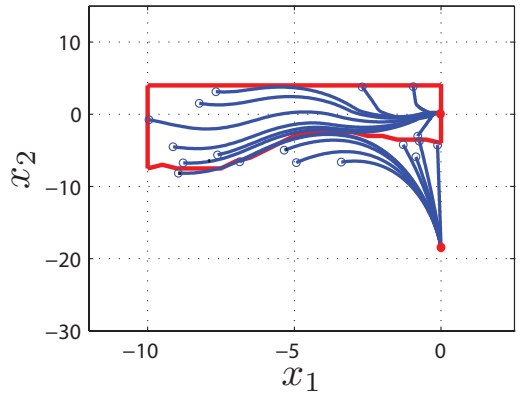

Fig. 13. Numerically estimated region of applicability of the System 4 (the red/black(in a black-and-white version) frame). An actual and spurious attractors are highlighted with circles. Note, the numerical method estimated a lower bound that goes along a trajectory with a good precision. Other bounds were left unchanged, i.e., in the other directions the considered region does not cross boundaries of the region of applicability.

will significantly depart from the real dynamics. This is due to the effect of the weights h_i associated to each Gaussian and how these influence the direction of the velocity vector: nearby the demonstrations, the influence of the closest Gaussian dominates that of all Gaussians, hence guiding closely the motion. However, far away from the demonstrations, the influence of all Gaussians becomes comparable and the resulting direction of velocity may point away from the signal. Generally our method allows to expand the region of applicability by at least a factor two over the initial volume which contains the training data.

As mentioned in the introduction, an inherent property of stable dynamical systems is their robustness to spatial and temporal perturbations. Figure 12 illustrates this aspect for one of the learned dynamical system, when the target is moved after the onset of the motion. As we see, the trajectories adapt smoothly to the change. Note, however, that the velocity profile may change abruptly when the perturbation occurs. To overcome this drawback it would be necessary to consider second-order dynamics.

As discussed previously, the GMMs encoding may result in spurious attractors outside the empirical stability domain C and in regions with low likelihood, see, e.g., Figure 11.

There are several reasons for the emergence of spurious attractors: first, the training set gives only a partial and noisy representation of the dynamics. Providing additional data in the regions around spurious attractors usually improves greatly performance. Second, the shape of the signal influence greatly stability. For instance, if the curvature of the trajectories changes smoothly, the spurious attractors, if any, will usually lie outside of the region of the confident inference, see Figure 11. However, if the system trajectories experience sharp changes in the curvature, as e.g., System 1, see the Figure 6, the likelihood of having spurious attractors in the considered region increases. By adding more Gaussians around the point with a sharp curvature one increases the guidance provided by the GMM and thus decreases the chances. By considering these practical shortcomings, one may improve a particular encoding to achieve the admissible performance.

TABLE IV
QUANTIFICATION OF RESULTS

System	Dim	Nb demonstrations	Nb of GMMs components	MPP [2]	MPP noise[1]	APP [3]	APP noise[1]	CPP [4]	CPP noise[1]	MPV[5]	MPV noise	APV [6]	APV noise[1]	CPV[7]	CPV noise[1]
System 1	2	6	7	0.08	0.60	0.006	0.08	4.24	12.01	0.69	1.10	0.003	0.019	3.43	7.11
System 2	2	6	2	0.01	0.09	0.008	0.03	5.93	14.50	0.02	0.08	0.001	0.003	1.02	2.02
System 3	2	6	6	0.03	0.11	0.007	0.03	5.33	14.21	0.17	0.22	0.008	0.01	5.85	12.01
System 4	2	6	13	0.01	0.22	0.004	0.01	2.05	11.2	0.06	0.21	0.003	0.007	2.41	5.41
System 5	3	10	12	0.07	0.12	0.01	0.10	6.12	16.53	0.21	0.31	0.006	0.013	6.51	18.04
KATANA experiment	3	4	4	-	0.34	-	0.21	-	20.08	-	0.17	-	0.10	-	13.52
HOAP experiment	3	4	5	-	0.42	-	0.33	-	26.02	-	0.21	-	0.18	-	15.60

[1] To estimate the accuracy of the proposed method in the presence of noise , we extend theoretical dynamics with a signal-dependent noise (see Figure 14) according to: $\dot{\xi} = f(\xi) + \eta(\xi)$, where $\eta(\xi)$ is a linear function: $\eta(\xi) = \sigma^2\gamma\xi$, where σ^2 is a variance of signal-dependent noise ($\sigma^2 = 20$ in our case), γ is a normal random variable with the zero mean and the unit variance.

Notation: R is a number of trajectories ($R = 20$ for estimating the precision of theoretical dynamics; R is equal to the number of demonstrations for the robotic experiments); Δ denotes to the length of a trajectory (e.g., $\Delta\xi_j$ is the length of a jth trajectory $\xi_j = \{\xi^i\}_{i=1}^{M_j}$, where M_j is the number of datapoints in a jth trajectory) $\hat{\xi}^i$ is a learned trajectory; ξ^i is a theoretical trajectory.

[2] MPP is a minimum positional accuracy: MPP= $\frac{\mathrm{R}\max_{i=1..M}\|\hat{\xi}^i-\xi^i\|}{\sum_{j=1}^{\mathrm{R}}\Delta\xi_j}$ specifies the maximum deviation of a reproduced trajectory from its precise theoretical value; MPP is normalized by the average trajectory length.

[3] APP is an average positional accuracy: APP= $\frac{R\cdot\sum_{i=1}^{M}\|\hat{\xi}^i-\xi^i\|^2}{M\sum_{j=1}^{\mathrm{R}}\Delta\xi_j}$. APP is normalized by the average length of all considered trajectories and by a number of datapoints in the considered trajectories. APP characterizes how far in average each reconstructed datapoint departs from its theoretical value in comparison with an average length of the trajectories, e.g. 0.01 means that in average a reconstructed position deviates by a fraction of 0.01 of the length of the trajectory.

[4] CPP is a cumulative positional accuracy: CPP= $\frac{\sum_{i=1}^{M}\|\hat{\xi}^i-\xi^i\|^2}{\sum_{j=1}^{\mathrm{R}}\Delta\xi_j}$. CPP is normalized by the cumulative length of the considered trajectories. CPP characterizes the cumulative average deviation of a reproduced trajectory from its theoretical value.

[5] MPV is a minimum velocity accuracy: MPV= $\frac{\mathrm{R}\max_{i=1..M}\|\dot{\hat{\xi}}^i-\dot{\xi}^i\|}{\sum_{j=1}^{\mathrm{R}}\Delta\dot{\xi}_j}$ specifies the maximum deviation of a reproduced velocity from its precise theoretical value; MPV is normalized by the average length of velocity trajectories.

[6] APV is an average velocity accuracy: APV= $\frac{R\cdot\sum_{i=1}^{M}\|\dot{\hat{\xi}}^i-\dot{\xi}^i\|^2}{M\sum_{j=1}^{\mathrm{R}}\Delta\dot{\xi}_j}$. APV is normalized by the average length of all considered velocity trajectories and by the number of datapoints in the considered trajectories. APP characterizes how far in average a velocity of each reconstructed datapoint departs from a theoretical value of velocity in comparison with an average length of the velocity trajectories.

[7] CPV is a cumulative velocity accuracy: CPV= $\frac{\sum_{i=1}^{M}\|\dot{\hat{\xi}}^i-\dot{\xi}^i\|^2}{\sum_{j=1}^{\mathrm{R}}\Delta\dot{\xi}_j}$. CPP is normalized by the cumulative length of the considered velocity trajectories. CPP characterizes the cumulative average deviation of a reproduced velocity profile from a respective theoretical profile.

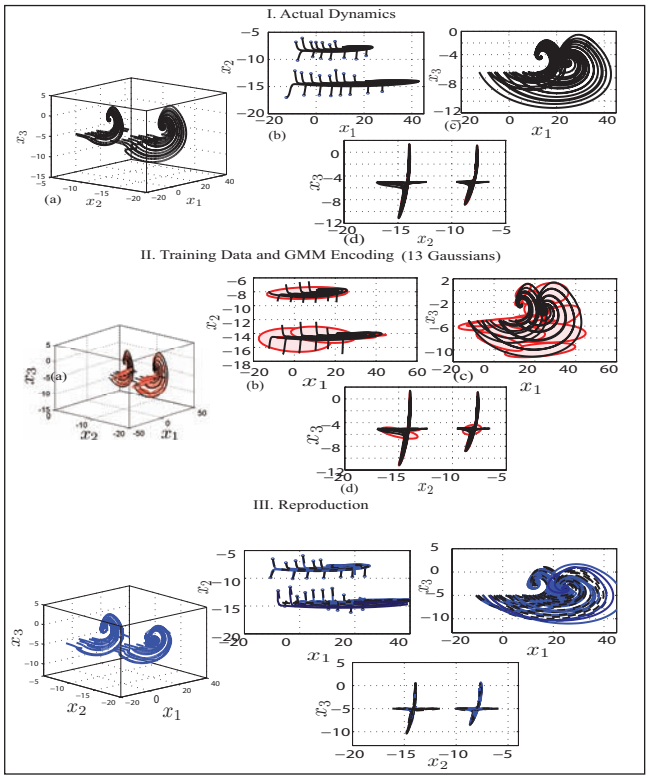

Fig. 15. Learning motion with two attractors. 3-dimensional trajectories are generated by *System 5* that displays a periodic behavior. Trajectories were demonstrated in the neighborhood of two asymptotically stable attractors. During the reproduction, the system managed to accurately reproduce dynamics around both attractors.

V. Application to Robot Control

Further, we validate the method to learn the dynamics of motion of a robot endeffector when trained through human guidance. Here, the dynamics of motion becomes the control law that iteratively moves the robot's arm along a trajectory.

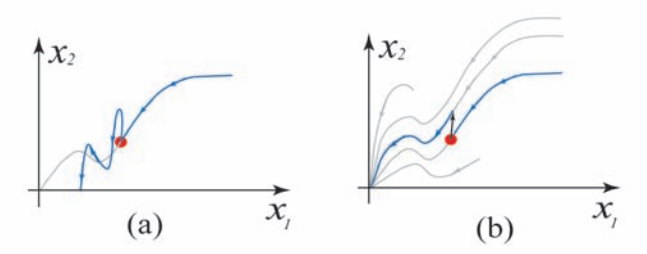

Fig. 16. (a) If a trajectory in the operation space passes through non-reachable joint positions IK may return velocity in the operation space that sends a robot too far from original trajectory, so linear assumptions of approximation of kinematics does not satisfy and overall trajectory tracking will fail. (b) In the case of motion encoding with a dynamical system, after perturbation the robot will not try to return to the previous trajectory violating the linear approximation of kinematics, instead the dynamical system will generate other trajectory from the point where the robot occurs.

A. Encoding motion in the operational space

Since the framework we defined above does not make any assumption as to the type of variables to be used for training, we are unconstrained in our choice of variables for controlling a robot. Here, we choose to describe motions according to the following variables: the translation component of motion of the end-effector is described by a vector of Cartesian coordinates $\mathbf{x} \in \mathbb{R}^3$.

Each demonstrated trajectory is, thus, represented by the following dataset: $D = \{\mathbf{x}_t, \dot{\mathbf{x}}_t\}_{t=1}^{M}$, where M is the number of datapoints in a trajectory. To reproduce a task, we first learn an estimate of the dynamical system using the method described in Section III-A and then use the Moore-Penrouse pseudo-inverse to compute the corresponding joint angles. Table V summarizes the steps of the reproduction algorithm.

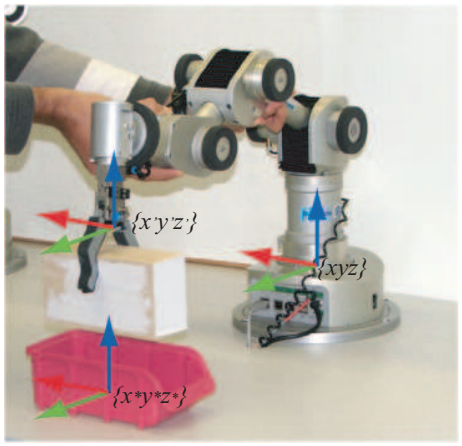

Fig. 17. We encode tasks in a referential located at the target and moving with it$\{x^*y^*z^*\}$; this referential is expressed in the fixed global referential $\{xyz\}$(usually we choose one attached to static parts of a robot). Actually, the motion of the robot end-effector is expressed as moving a referential associated with the end-effector $\{x'y'z'\}$.

TABLE V
On-line task reproduction

1	Assume that a controller $\hat{f}_{\mathbf{x}}$ has been learned, the robot is thus ready to reproduce a task
2	Detect a target position in the global referential $\{xyz\}$; see Figure 17: $\mathbf{x}^*$
3	Recompute the current position of an end-effector in the target referential $\{x^*y^*z^*\}$: $\mathbf{x}_0$
4	**LOOP** until the target position is reached
5	infer the velocity for the next iteration t through GMR Eq.9: $\dot{\hat{\mathbf{x}}}_t$ $\dot{\hat{\mathbf{x}}}_t = \sum_{k=1}^{K} h_{k,\mathbf{x}}(\mu_{k,\dot{\mathbf{x}}} + \Sigma_{k,\dot{\mathbf{x}}\mathbf{x}}\Sigma_{k,\mathbf{x}}^{-1}(\mathbf{x} - \mu_{k,\mathbf{x}}))$
6	solve the Inverse Kinematics problem to find: $\dot{\mathbf{x}}_t, \dot{\boldsymbol{\theta}}_t$
7	compute a new position $\mathbf{x}_t, \boldsymbol{\theta}_t$
8	**END**

B. Set-up

We validated the above method in three practical tasks, see Figure 18, 23. We also implemented the theoretical 3-dimensional *System* 5, as a motion generation policy for the robot. To highlight the generic character of the approach we ran experiments with three different robotic platforms: a 6 degree of freedom industrial-like KATANA arm from

Neuronics, a 4 degree of freedom robot arm of the humanoid robot HOAP-3 from Fujitsu, and a 7 degree of freedom humanoid platform i-Cub which 7 degree of freedom of the right arm have been used see Figure 23-(a).

For KATANA and Hoap demonstration is accomplished through kinesthetic teaching. In the case of iCub, as the motors are not back-drivable, demonstration is accomplished via teleoperation of the robot arm by a human teacher. The simultaneous control of all 7 degrees of freedom is conducted through a joint recording system placed on the human. A mapping from human to robot arm allows the human to directly control the motion of the robot arm, by moving his own arm. Measurements from the motion sensors are mapped in real-time into the robot joint commands, therefore, the human teacher is getting immediate visual feedback regarding accuracy of demonstrations he/she provided. While moving the robot records observations, taken from its own sensors. In detail, sensing units from commercial $XSens$ joint recording system are placed on the upper and lower arm, and back of the palm, of the human; see Figure 23. During the reproduction i-Cub was controlled in real-time at the frequency of 50Hz. An external color-blob tracking vision system was used to detect the position of the ping-pong ball.

C. Experiments with KATANA

The first experiment consists in the KATANA putting an object into a container. Here, the KATANA arm was taught to put a rectangular wooden brick into a rectangular container; see fig.18-left.

In the second experiment, the KATANA was controlled with *System 5* with the origin of the system positioned on an arbitrary object. This experiment meant to test the ability of the learned system to *generalize* to context unseen during training and to quickly adapt to perturbations.

This experiment meant to show that the theoretical dynamical system could be of practical use to guide robot motions for a simple reaching task. It also demonstrate that, as shown in simulation, the system is stable and follows the trained (and known) dynamics of motion.

D. Experiments with HOAP-3

The clench of the HOAP-3 is rather small, therefore it can grasp only thin objects. In this task the robot had to grasp a box which is thin along one dimension, so the robot should follow a specific path to properly position its hand; see fig.18-right.

During training, the robots were shown the tasks 5 times by a human user guiding their arms. Values of the robots joints were recorded during this passive motion and used for reconstructing the position of the end-effector.

E. Experiments with iCub

The experiments with iCub aim at demonstrating the abilities of the proposed approach (1)to *generalize* to unseen conditions (i.e. with the robot's end-effector starting from different locations and with the target at different locations than these seen during training), and (2) to adapt to temporal perturbations (extended or shortened duration of motion). To emphasize the importance of being time-independent and representing the motion in the state-space (for tackling both spatial and temporal perturbation (see discussion in Section II.B and C) we compare performance when trajectories are encoded with the proposed approach and when these are encoded using Dynamic Movement Primitives[Ijspeert et al., 2001, Pastor et al., 2009, Hoffmann et al., 2009] (the sole approach to motion representation with dynamical systems which is shown to be asymptotically stable at the target; see also Introduction).

In these experiments we also focus on the comparison of our with the other dynamical system approach to motion generation Dynamic Movement Primitives . In this experiment the iCub learn to reach a ping pong ball with a forehand motion but stopping at the target. Note that this experiment on purpose replicates the task of reaching for a ball with a tennis racket in the original DMP paper [Ijspeert et al., 2001] (there the velocity was also zero at the target). It is still relevant for comparison, as allows to highlight different aspects of the algorithm that are advantageous in the comparison to the existing learning approaches.

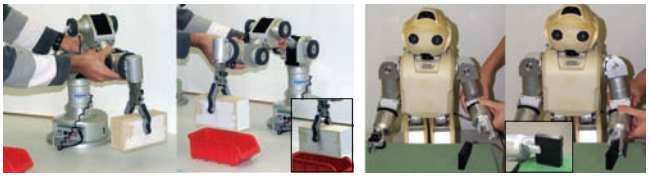

Fig. 18. Set-up of the experiments. Left: KATANA puts a wooden brick into the container, to achieve the task the robot should lift the brick and move it following an elevating trajectory. Right: HOAP-3 grasps a box, to achieve the task HOAP should approach the box with a specific orientation and than lower its arm, as the clench is small, see small figure in the corner.

F. Results of Learning Dynamics from Motion Data

After training, we tested the system by requesting the robots to reproduce the tasks in various conditions. The results of the experiments are summarized in Figures 20-19. To test

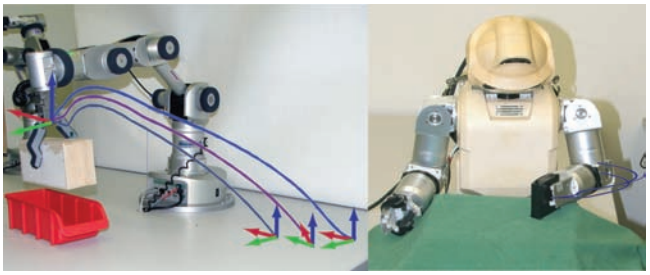

Fig. 19. The results of reproduction of dynamically generated trajectories on the robots. To check the generalization abilities of the learned dynamics the trajectories were reproduced from different initial positions.

the generalization abilities and the stability to perturbations we performed experiments by changing the starting positions of the robots and shifting the container (for the KATANA's

experiment) or the box (for the HOAP-3's experiment). Results are presented in Figure 19; in both experiments learning of position control was successful and the robots all reached successfully the targets and accomplished the tasks. Results

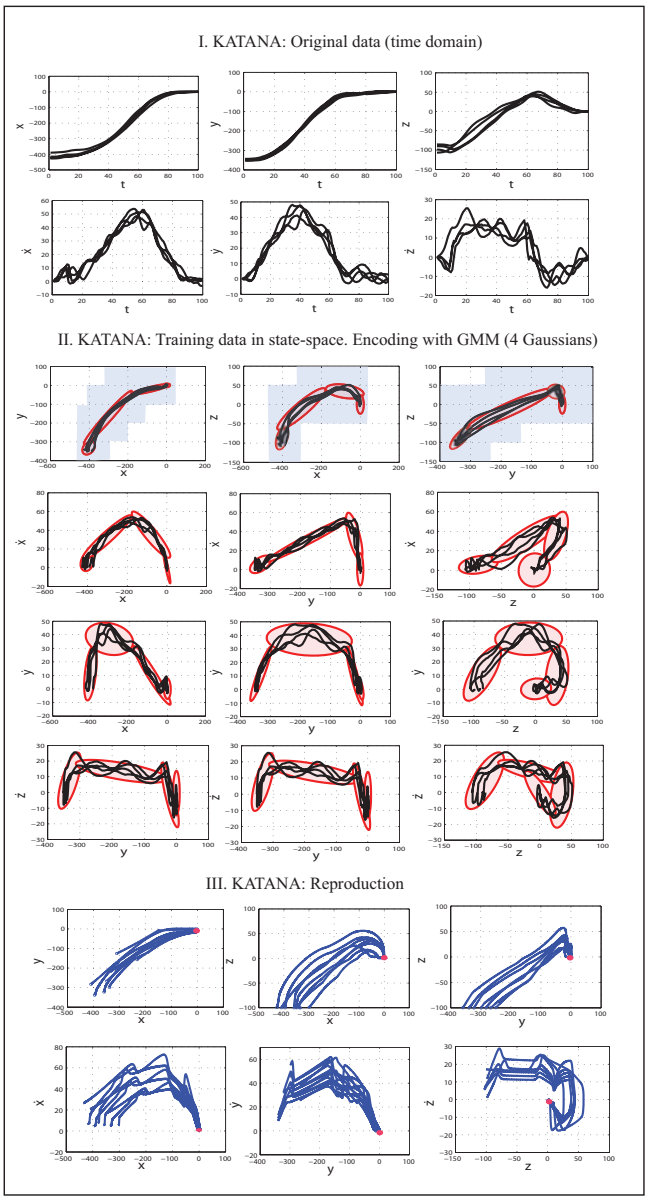

Fig. 20. KATANA experiment 1: Results of encoding and reproduction of the experiment where KATANA had to put a brick into a container.

of generalization for the second experiment with KATANA reproducing *System 5* are presented in Figure 21 - II. The area where demonstrations were provided is depicted in Figure 21 - II (b) with red squares. The system further allowed to reproduced the motion starting from any position of the sub-space of the workspace, depicted in grey. Note, that even few demonstrations provide good generalization properties. The ability to generate a trajectory from an arbitrary initial position to the target with a relevant velocity profile is a strong point of encoding motion with Dynamical Systems in the state-space, furthermore it provides real-time adaptation to perturbations in the position of the target. The Figure 21-I presents results

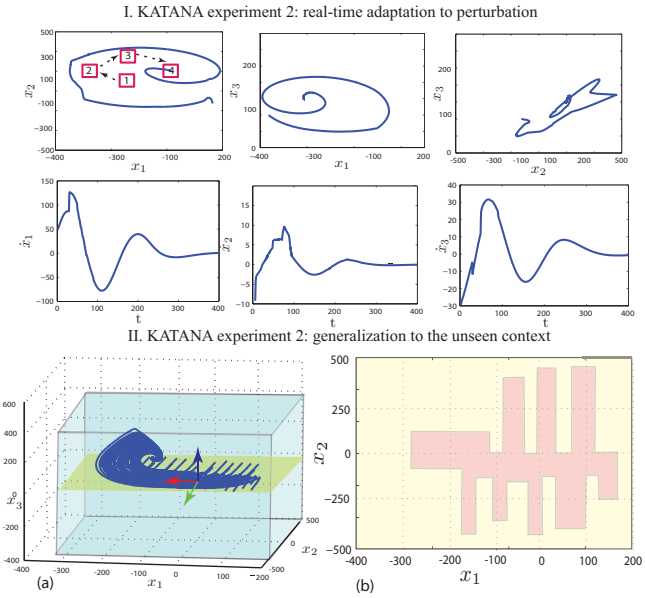

Fig. 21. KATANA experiment 2: *I. Real-time adaptation to perturbations*. The target was shifted several times from the position 1 to the position 4. First row: trajectory of the robot's end-effector; second row: velocity profile. *II. Generalization to the unseen context*. (a) The approximate workspace of KATANA is highlighted by the blue box (light grey in a black-and-white version), the reproduction was systematically tested starting the robot from positions on the starting plane (yellow/darker grey). (b) The robot was required to reproduce the motion from points monotonically covering the yellow/light-grey sub-part. For comparison the part of space where the demonstrations were provided is in pink/dark-grey. Note, the demonstrations are sparse, but the system manages to generalize to other parts of the workspace.

of tracking a marked object mapped into the attractor of the dynamical system. After shifts of the target, the robot finally reaches the object following the demonstrated position and velocity profile.

G. Comparison with Dynamic Movement Primitives

In addition to the theoretical comparison we provide in Appendix I, we performed an experimental comparison of the performance of our approach with that of Dynamic Movement Primitives (DMP)[11]. DMP represents one of the first examples of the strength of dynamical system encoding in providing a flexible framework for learning arbitrary non-linear motions. The strength of DMP and of its most recent improvements [Pastor et al., 2009, Hoffmann et al., 2009] lie in that DMP can fit in an arbitrary non-linear system from a single demonstration. The method proceeds by modulating a *linear* stable dynamical system with an acceleration profile learned from a single demonstration. The modulating function is dependent on the internal clock (the canonical variable s; see also Appendix I). This implicit time dependency makes the system sensitive to perturbation as we demonstrate here. Note that even recent improvements offered on the method in [Pastor et al., 2009, Hoffmann et al., 2009] do not resolve this time-dependency issue that we tackle in our approach. In addition, high accuracy is counterbalanced by DMP's inability to generalize well

[11]The MATLAB code implementing both methods as well as video materials of the comparison are available at http://lasa.epfl.ch/elena/learning-dynamics.htm

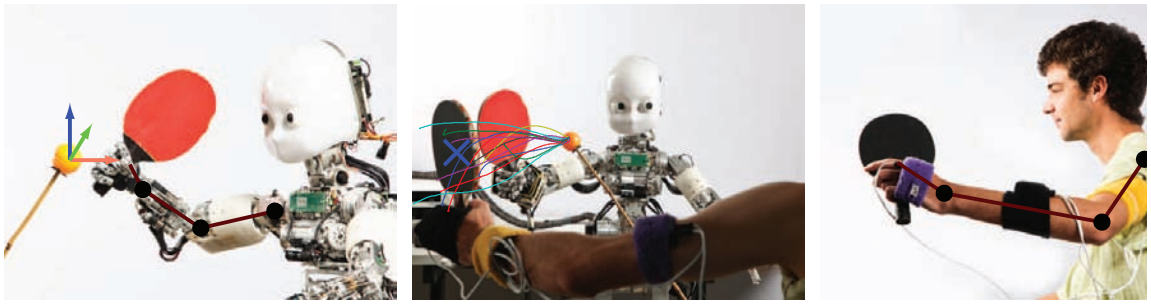

Fig. 23. (a) The humanoid robot iCub we use in the experiments. We encode tasks in the frame of reference located and moving with the target; this referential is expressed in the fixed global frame of reference (usually we choose one attached to static parts of a robot). (b) A human teacher demonstrates a ping-pong motion to the robot. (c) Three XSens motion capture sensors are attached to the hand, forearm and upper arm of the demonstrator and allow to reconstruct the motion of each joint

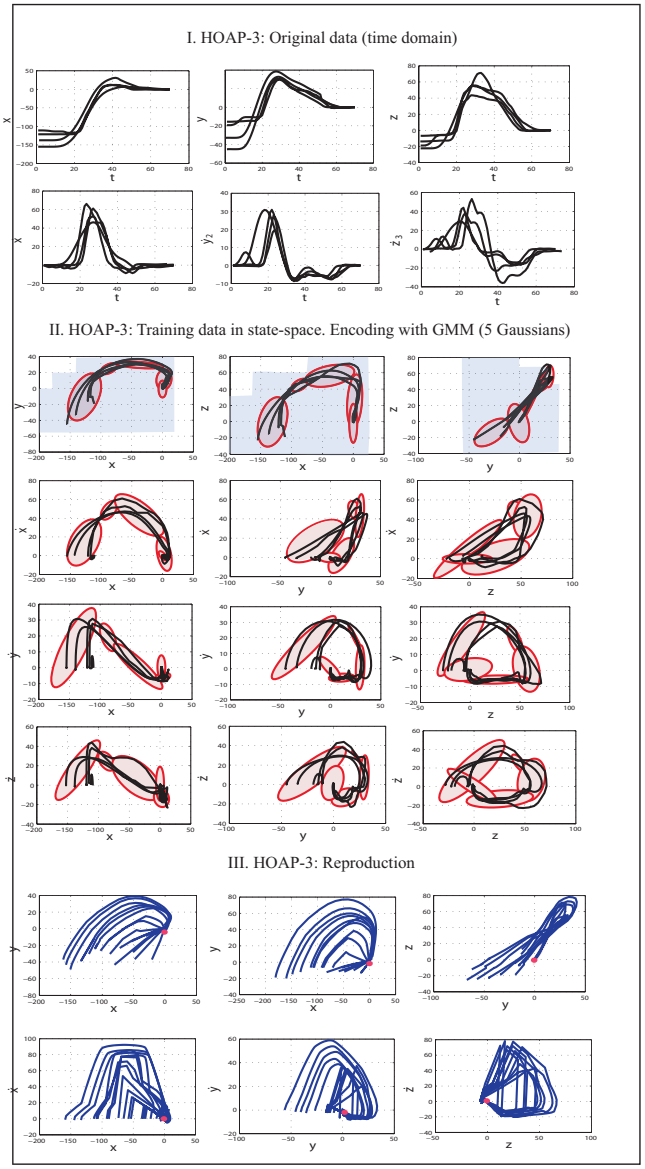

Fig. 22. HOAP-3 experiment: Results of encoding and reproduction of the experiment where HOAP-3 had to grasp a box.

outside the demonstrated trajectory. Our method offers a true time-independent encoding and generalizes outside the area covered by the demonstration.

For illustrative purposes, we first assess differences by learning a theoretical 2D dynamics and further compare the performance of the methods in the ping-pong task. Before turning to the actual comparison, we shall highlight theoretical differences between the two methods and motivate our choice of qualitative criteria for comparison.

In DMP a modulating function $\hat{f}(s)$ is learned either using LWR [Atkeson and Moore, 1996] or LWPR [Vijayakumar and Schaal, 2000] from a *single* demonstration (see discussion concerning combining several demonstrations for learning DMP in Section V-G4). In contrast, the approach proposed here learns a model of motion dynamics using *several* demonstrations encoding these in the *state-space*. The latter is particularly a fundamental difference, since with state-space, encoding one immediately gets a feedback signal that allows to adapt a velocity profile directly based on the current position of the robot. Furthermore the learning procedures in DMP and our approach differ. Since DMP uses a single demonstration to train the system, if the data contain noise it will be fitting noise as well as the true signal. Combining several demonstrations allow our method to build more accurate estimates of an actual underlying dynamics [Coates et al., 2008]. However, this does not necessarily lead to the exact trajectory fitting if the data are noisy.

Therefore, next, we compare the two methods in terms of their qualitative performance in the case of (1) changing the initial position (generalization to the unseen context); (2) spatial perturbations (changes in the target position *after the onset* of a motion); (3) temporal perturbation (changes in the target position *after the onset* of a motion that considerably *change the time* of reaching the target).

We follow the most recent formulation of DMP from [Pastor et al., 2009]; see also Appendix I. The coefficients D and K are chosen to guarantee critical damping. Note that we have implemented DMP as describe in [Pastor et al., 2009], i.e. without a heuristic to re-index the canonical variable s so

as to handle perturbations[12].

1) Comparison of generalization abilities: The generalization abilities of DMP are limited; since the representation contains no information about dependencies between position and velocity (or acceleration), when trajectories are to be reproduced when starting from unseen parts of the workspace they become *scaled versions* of the demonstrated trajectory. In the case depicted in 25-(b) we see an example of reproduction of the motion with the onset of the motion located in the middle of the original demonstrated trajectory. Instead of reproducing the remaining part of the original trajectory, DMP reproduces the whole trajectory, scaling it so to fit into the distance to the target. Our system, in contrast, follows the relevant segment of the demonstration. The scaling effect is even more noticeable in the three-dimensional case of the ping-pong task; see Figure 26.

Sole scaling of trajectories instead of following an actual dynamic pattern can fail reproduction: one of the reasons why humans depart from straight-line trajectories consists in an intention to satisfy external constraints, e.g. geometrical constraints of manipulated objects. The trajectories provided by a human are, therefore, implicitly encode these constraints in the form of a specific curvature [Petreska and Billard, 2009]. An efficient motion representation should be able to encode these constraints and allow for their faithful reproduction in relevant regions (e.g. around the target). Note, the demonstrated trajectories in Figure 25-27 are strongly curved around the target, this can be, for instance, due to the presence of an obstacle that should be avoided or such a path might be dictated by the particular shape of a manipulated object. Therefore, it is crucial for the robot to satisfy this pattern during reproduction. The trajectories generated by our system follow the approach direction that has been demonstrated, while DMP runs considerable risk of violating implicit constraints and bumping into obstacles.

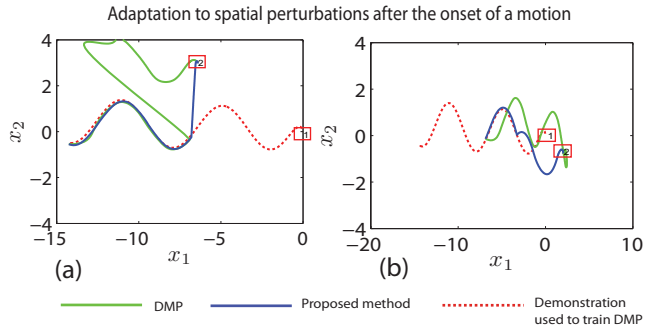

Fig. 25. Comparison of adaptation to spatial perturbations of the proposed method and DMP [Hoffmann et al., 2009], learning a theoretical noise-free dynamics. Due to scaling that DMP performs for adapting a learned acceleration profile to the conditions after perturbation, a generated motion may have an unexpectedly excessive curvature (a) or can overshoot the target (b).

2) Comparison of robustness to spatial perturbations: Here, we compare our method in terms of robustness to spatial perturbations, i.e. to displacement of a manipulated object or end-effector occurring after the onset of a motion but that do not cause a delay in the time required to reach the target. Since both DMP and our approach guarantee asymptotic stability, here we focus on other qualitative aspects of robustness. Namely, we look at whether both systems can reproduce key characteristics of the motion, such as the curvature, in both the noise-free model and in the ping-pong task, see Figure 25 and 26 respectively. DMP does not adapt the shape of the trajectory when moved to an arbitrary location in the workspace. This hence can lead to too curved motions and overshoots at the target.

[12]The authors suggest that it is sufficient for adaptation to move the target position g

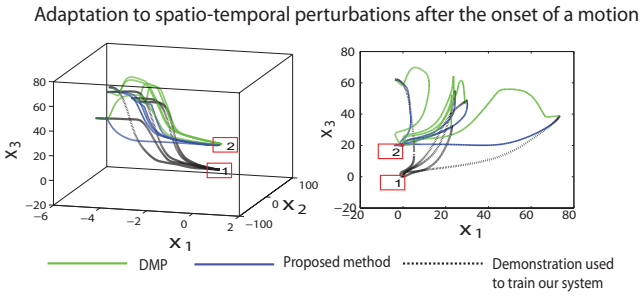

Fig. 26. Comparison of adaptation to spatio-temporal perturbations of the proposed method and DMP [Hoffmann et al., 2009] in the ping-pong experiment. The ball has been moved up (from position (1) to position (2)) after the onset of the motion, DMP trajectories produce strong swings and tend to overshoot the target.

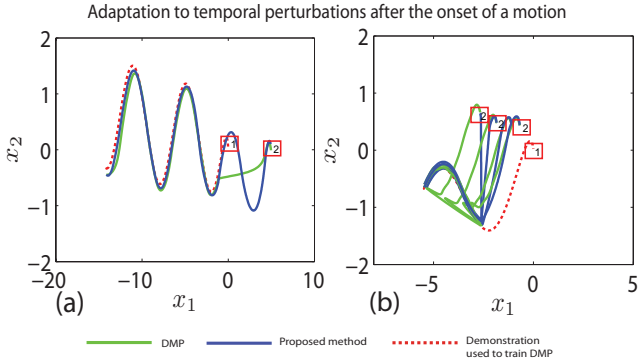

Fig. 27. Comparison of adaptation to temporal perturbations of the proposed method and DMP [Hoffmann et al., 2009]. The target has been shifted so that the duration of motion has been increased (a) or decreased (b). (a) The target is moved from position (1) to position (2) which is farther from the robot's end-effector, DMP takes the shortest path to the initial position of the target and stretches it to reach the shifted target position, this results in an almost straight line trajectory which potentially may violate external constraints implicitly encoded in the demonstrations. (b) The target shifted so as to decrease the duration of motion, in this case DMP scale the trajectory and produce the jerky motion right after the perturbation; in this case DMP require more time to reach the target than our system.

3) Comparison of robustness to temporal perturbations: Here, we consider perturbations in the target position after the onset of a motion that result in a considerable delay to reaching the target. Results are shown in Figures 26, 27. In Figure 27-(a) the target is moved from position (1) to position (2) which is *farther* from the robot's end-effector, DMP takes the shortest path to the initial position of the target and stretches

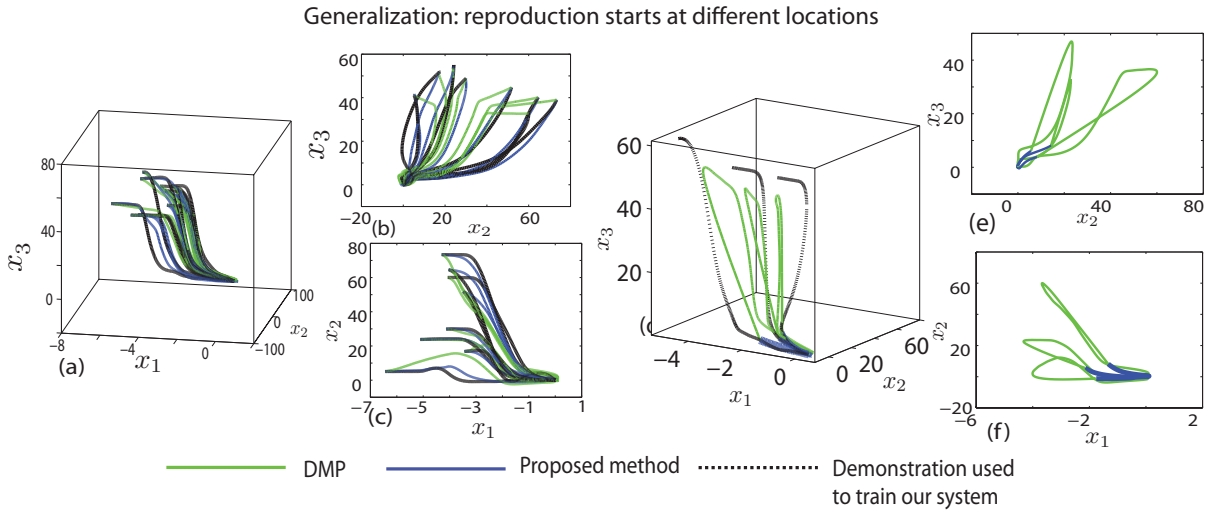

Fig. 24. Comparison of generalization abilities of the proposed method and DMP [Hoffmann et al., 2009] in the ping-pong experiments. Here, due to the noise, our system tries to extract a common noiseless pattern and therefore reproduction does not follow the demonstrated trajectories exactly along the whole motion (as in comparison to the noise-free case; see Figure 9). However, in comparison with DMP generated trajectories, the trajectories of our system exhibit more similarity (in terms of the shape of trajectories) with demonstrations, particularly starting from unseen positions where DMP produces unexpected swings.

it to reach the shifted target position. This results in an almost straight line trajectory which may violate external constraints implicitly encoded in the demonstrations. The deformation of a motion pattern also occurs in the case of the ping-pong experiment when the ball is moved away from the robot; see Figure 26. DMP fail to reproduce the demonstrated slope of trajectories; see particularly Figure 26-(b).

In Figure 27-(b) the target is moved from position (1) to position (2) which is *closer* to the robot's end-effector, DMP try to fit the learned trajectory into the new spatial conditions which results in jerky motion. Our system drives the trajectory directly to the target, so the robot reaches the target faster than with DMP.

4) Conclusion of comparison: DMP provide a light and elegant tool for learning a stable estimate of a motion dynamics from a single demonstration. It allows for adaptive scaling of a demonstrated acceleration profile and ensures asymptotic global convergence to the target. This tool is particularly useful if the robot is supposed to generate a trajectory starting from a neighborhood of a demonstration, and if one requires an accurate replication of the demonstrated path when starting in the original configuration; see Figure 28.

However, this solution comes with the drawbacks: depending on the starting position and perturbations along a motion, a resultant trajectory might not satisfy implicit constraints encoded in the demonstrations. As DMP do not address learning the dependencies in the state-space, the temporal robustness cannot be guaranteed as showcased in the comparison.

The implicit time dependency of DMP makes the system sensitive to temporal perturbations, as we show here. Note that this implicit time-dependency remains even in the recent reformulation of DMP suggested by [Hoffmann et al., 2009, Pastor et al., 2009, Park et al., 2008]. The time dependency is conveyed through the canonical variable, that acts as a clock for the system[13]. To adapt to changes in the motion's duration associated with different initial positions or significant spatial perturbations, one must use a heuristic to re-set the canonical variable. Failing this, the canonical variable forces the modulation term to reproduce the same acceleration profile irrespective of where in the workspace the robot starts to move or where a perturbation occurs. Furthermore, once the canonical variable decays, it ultimately cancels the modulation terms. As a result, the system is then driven solely by a linear dynamical system (governed by the first two terms of the right hand-side of Equation (A-I-3) of Table VI). Therefore, this dependency on the phase variable may result in undesirable behaviors which were highlighted above, see Figure 24-25.

These undesirable responses of the system would be avoided if one is able to find a way to rescale the phase variable. It is however not easy to determine a robust heuristic for inferring the optimal phase if the motion duration is unknown, e.g. after perturbations. In the method we propose here, this time dependency is removed entirely, hence avoiding the problem of finding a heuristic. DMP were a major step in introducing DS as a robust method for robot motion generation and the reformulation proposed in [Hoffmann et al., 2009, Pastor et al., 2009, Park et al., 2008] offers an elegant method to perform obstacle avoidance, a problem to which we do not offer a solution here.

As discussed above, the method we presented here ensures local asymptotic stability at the attractor (as per *Definitions* 6 and 7 of Table I), within the region of attraction. A secondary mechanism should be employed to stop the robot if a perturbation sends it outside of this region. Alternatively, one could also assume that, outside of the region of applicability, the motion is driven by the linear globally stable dynamics

[13]Specifically, this is the variable θ in Equation 11 of [Park et al., 2008], which is equivalent to the variable s in the original DMP formulation [Ijspeert et al., 2001] and its another reformulation [Hoffmann et al., 2009], see also Table VI of our manuscript.

defined by the last Gaussian. DMP and its extensions in contrast have the nice property of being globally stable. Note that, in our experiments, the region of attraction was large, covering about half of the workspace of the robot, see Figure 20-22. Furthermore, from a practical point of view, ensuring only local stability in many cases is not so restrictive. Non-linear motions are often driven by local constraints, e.g. the shape of manipulated objects, and it may make little sense to arbitrarily reproduce these constraints in the whole workspace. Statistical learning is local by nature; hence, one cannot ensure that inference far from the demonstrations will be relevant in a statistical sense (as the likelihood of the data will be negligible): Figure 29 shows an example of motion generated with DMP, where the initial positions are located far from the original demonstrations.

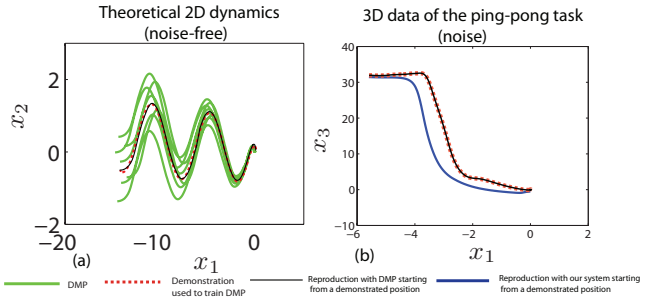

Fig. 28. (a) Illustration of the high accuracy of DMP at reproducing trajectories that start in a small neighborhood of a demonstrated trajectory. When starting the motion at the same location as that demonstrated, reproduction fits the original signal very accurately. (b) When reproducing the noisy training data from the ping-pong task. DMP accurately fit each demonstration separately. This leads to over-fitting as each trajectory contain noise inherent to the physical world.

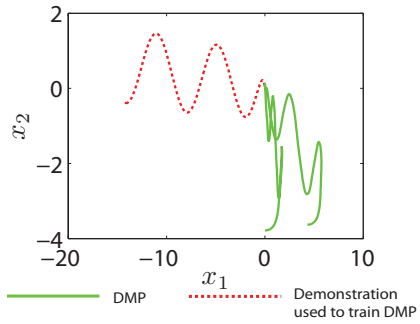

Fig. 29. An example of a trajectory generated with DMP and starting far from an original demonstration. Note, that although the motion is globally asymptotically stable, the resulting trajectory makes little sense.

VI. DISCUSSION AND FUTURE WORK

For scientific completeness, we now revisit each of the hypotheses underlying our approach and suggest alternatives solutions.

A. Multi-dimensional systems, first order dynamics

The method proposed here allows learning of non-linear multivariate dynamics where the correlation between the variables is important. Other works on dynamical control consider each degree of freedom separately, hence discarding information pertaining to correlation across the joints. While storing correlations across the joints is costly (in GMM, it forces one to compute the complete covariance matrix, rather than computing only the diagonal elements), it is advantageous as correlations contain features characteristic of the motion. For instance, in bimanual coordination tasks in which left and right arms should follow different dynamics while doing so in coordination [Gribovskaya and Billard, 2008], embedding the correlations in the representation ensures the reproduction of both the dynamics of each arm and the correlations across the arms. Furthermore, learning correlation between a multivariate signal and its derivatives allows to considerably decrease a number of Gaussians required to accurately encode the training dataset. While we started with the hypothesis that the control law followed a first order dynamics, the method proposed here may be extended to learn higher-order dynamics (as higher-order systems can always be expressed in the canonical form as a set of first-order systems). That is particularly relevant for applications where it is necessary to control the acceleration profile. We intend to address this problem in future work.

Potential difficulties concerning shifting into higher-order derivatives that can be envisioned, are associated with the increased dimensionality of a resultant statistical problem. With an increase in the number of dimensions, a stable approximation would require more training data or need to introduce certain heuristics to partially decouple the problem into a set of systems with lower dimensions.

B. *Time independency* vs *time dependency*

In this paper, we advocate that time-independent encoding in the state-space offers more robust representation in comparison to traditional time-dependent encoding. Results confirmed that for a certain range of motions, the state-space representation is indeed highly robust to spatial and temporal perturbations. Moreover, it allows to reproduce tasks even in unseen parts of the workspace.

Yet, certain motions, such as those requiring the synchronization with an external dynamics, should be encoded using a time-dependent representation or, if the external dynamics is known, using an explicit parametrical coupling of two time-independent dynamics, such as that done in [Ijspeert et al., 2001]. Another limitation of the time-independent representation relates to the possibility of encoding compound motions: in this case, the whole motion may be segmented into a set of simpler ones governed by a single attractor. However, the problem of how to transit across these systems remains an open issue.

C. Kinematic controller

In the experiments reported here, control of the robot was purely kinematical, encoding the desired kinematic trajectories, but not taking into consideration the dynamical properties (actual torques) of the robot limbs. An additional control step was then necessary to convert positions into motor commands

by means of the inverse dynamics (KATANA) or a PID controller (HOAP-3).

We should emphasize that the proposed method can be coupled with operation space control [Khatib, 1987, Hsu et al., 1989, Nakanishi et al., 2005]: operational space control aims at executing trajectories or forces defined in the task space of an end-effector. In this framework the general problem consists in finding a control law which governs the robot along a joint space trajectory such that the controlled element (e.g., end-effector) follows a desired kinematic trajectory or force profile in the task space. Our method provides an input for operation space control by generating the desired kinematic trajectory in real-time.

Learning the inverse dynamics and operational space control [Peters and Schaal, 2008], while a highly valuable topic in itself, is beyond the scope of the present paper. Further, considering that many of the current robotic platforms are controlled in joint position or velocities, the proposed approach combined with the inverse kinematics is thus valid for a large set of applications.

The proposed approach essentially compensates for the robot's hardware limitations (joint velocity and torque limits) that can lead to deviations from original commands: e.g, if a robot is not able to reach a particular position in a given time span due to angular velocity limits, the system at each time step will recompute the next motor command based on an actual position of the robot. Therefore, while the hardware limits can slow down the motion, but the real-time dynamical controller still allows the robot to follow a desired path.

A problem of the overall stability of a system consisting of a low- and a high-level controllers may arise if the low-level controller does not support a control frequency necessary for the high-level controller to be stable: if the frequency of the low-level controller is too low, the dynamical planner at the high-level will tend to overshoot a target and may fail to converge. However, the state-of-the art robotic platforms operate at a frequency that is sufficiently high to dynamically generate a stable trajectory given a stable dynamical controller at the high-level.

D. Choice of statistical framework

GMMs being a global statistical techniques (by opposition to local non-parametric methods such as LWPR, GPR) was shown to be suitable for estimating dynamics from sparse demonstrations, that are typical of programming by demonstration applications. However, neither GMMs nor LWPR and GPR ensure stability of a learned approximation. Here, we proposed an algorithm that leads to local asymptotical stability and gradually improves the quality of the approximation while widening the region of applicability C. Potentially, the same procedure may be adopted for other statistical frameworks. However, the accuracy of the approximation may significantly vary depending on a particular choice.

One should note that EM is more computationally expensive than LWPR with a number of iteration steps during training of $O(K \cdot M \cdot N)$ in comparison to $O(N)$. Both of these however remain small in comparison to GPR. Similarly to LWPR and in contrast to the GPR-based methods, GMR's computational costs for the retrieval procedure are low and increase linearly with the number of parameters. Additionally, GMM-based models result in much less parameters due to the coarse representation.

A part of computational complexity of the proposed method comes from the iterative estimation of the region of applicability, which requires $n_1 \cdot .. \cdot n_N \cdot M$ iteration steps ($n_1..n_N$ are the respective sizes of the mesh along the N dimensions, M is the number of data points). In our experiments, estimation of the region of applicability hasn't exceeded 100-120sec. Since learning can be performed offline and the learned model allows reproduction of a task without any additional computation such high computational complexity is counter-balanced by the low computational cost during the retrieval.

E. Real-time adaptation to perturbation vs traditional planners

One of the strengths of the proposed approach is its ability to cope with perturbations in real-time. By perturbation we referred to unexpected changes in the positions of the attractor or of the robot's joints during motion. We demonstrated how the learned dynamics with a position of an object mapped into an attractor can successfully track the object. Such a flexibility combined with the guarantee of ultimately reaching the object is one of the major advantages of the proposed method in comparison with traditional planners [Yokoi et al., 2009, Yoshida et al., 2008, Diankov and James Kuffner, 2007, Kuffner et al., 2002]. One should emphasize that planners, in turn, are advantageous when the environment is known and for providing mechanisms for obstacle avoidance. The latter is, however, achieved by introducing a heuristic-based cost function that penalizes certain directions. Potentially, our approach may be combined with such a cost function that perturb an output of a learned dynamical system pushing it away from obstacles.

Note, that our system though introduces certain hypotheses, still remains rather generic regarding tasks it may reproduce, furthermore, it may work with limited and inaccurate information about the environment, as it does not require any costly replanning. At the same time, to benefit from optimal planning and capacity for obstacle avoidance, one should provide an algorithm with precise information regarding objects in the workspace and introduce certain task-related heuristics to improve convergence.

F. Learning whole-body motions in high-dimensional spaces

The complexity of learning grows with the number of degrees of freedom: building an accurate model of the motion of each DOF separately and in correlation would require a considerable amount of training examples. This problem is shared a fundamental problem that affects all statistical learning approaches (though there are many ways that have been proposed to reduce the effect of high dimensionality on computational costs, this remains a fundamental issue); it is likely that nature has taken ways to solve this problem

by segmenting it into smaller problems controlling for less degrees of freedom at a time.

Two observations from human motion control may give insights into possible ways of tackling multi-degree of freedom motions. First, the human motor system tends to decouple control of different degrees of freedom into control of sets of degree of freedom in lower dimensions [d'Avella et al., 2003, d'Avella and Tresch, 2002]. Second, within each set of coupled degrees of freedom, each particular degree of freedom is not controlled independently, but rather in the synergy with others [Kelso, 1995, Giszter et al., 1993]. Finding a way to apply these hypotheses for robot learning and control would significantly improve both the efficiency and the quality of learning of complex coordinated motions.

G. *Single* vs *several attractors*

A further hypothesis pertaining to the work presented here was the idea that the dynamical system to be discovered had a single or several known fixed point attractors. This can be considered as a limitation, as a dynamics may be governed by the existence of more complex orbits than merely fixed points. For example, an arbitrary free motion may have a particular curve in space as attractor. The applicability of the proposed method in this case will mostly depend on the quality of training data; further no stability can be guarantee. Procedures for ensuring stability of complex orbits may substantially widen the class of motion under consideration, covering dancing or sport motions that are usually characterized by the existence of certain curves to which all trajectories converge.

H. *Training data*

The generalization properties of dynamical controllers directly depend on the quality of training data; the aspect common to all statistical learning methods. It might be compensated in different ways: 1) by providing an exhaustive set of accurate demonstrations; 2) by allowing a robot to explore on its own (considered in Reinforcement Learning [Guenter et al., 2007]); 3) by providing more variability in a limited set of demonstrations (the problem has been discussed in [Calinon and Billard, 2007]). The first option does not agree with a requirement of user-friendliness of teaching interfaces, as a number of demonstrations should be kept bearable for a user; the second approach may require additional time; therefore, we concentrate on improving quality of demonstrations by introducing more variability into a small set of demonstrations.

I. *Kinesthetic teaching*

For demonstrating tasks we used the kinesthetical teaching approach that consists of directly demonstrating the task using a robot's own body. One of advantages of this approach is that the human can feel limitations of the robot's architecture and adapt his/her intuition about an optimal or efficient motion accordingly. Although we actively exploit this learning paradigm, other approaches such as vision-based learning are also widely used and can be more intuitive for humans. Our system may be applied to the motion data obtained through different modalities.

J. *Practical consideration*

From a practical point of view, mapping position of manipulated objects into attractors of Dynamical Systems considerably improves the precision of motion at a target and therefore allows considering prehensile tasks in the framework of Programming by Demonstration; where so far generation of large-scale motions has been addressed.

The approach was shown to be generic in that it did not depend on the particular geometry of the robot's arm, nor on the particular variables to be learned. Indeed, it could be successfully implemented to control robot arms with different geometries and for learning the dynamics of different variables inherent to position and orientation control. The MATLAB code and supplementary material are available at http://lasa.epfl.ch/elena/learning-dynamics.htm

VII. Summary

In this paper, we proposed a method for learning a nonlinear multi-dimensional dynamics of motion through statistically encoding demonstrated data with Gaussian Mixtures. Further, we addressed the problem of ensuring stability of a resultant control law: first, we formulated conditions that parameters of GMMs should satisfy to guarantee local asymptotical stability of an attractor, then we proposed a numerical procedure to verify boundaries of the region of applicability where the control law can be securely applied.

To test the method, we conducted two types of experiments: 1) learning theoretical dynamics with known mathematical forms to estimate the accuracy of approximation and 2) learning dynamics of manipulation tasks recorded with two different robotic platforms to assess the applicability of the approach to the noisy data. In all experiments the system demonstrated good results in terms of the high accuracy during reproduction, ability to generalize motions to unseen contexts, and ability to adapt on-the-fly to spatio-temporal perturbations. We also showed how the system can encode more than one attractor, successfully reproducing each separate dynamics locally around each attractor and separating the flows leading to the different attractors.

VIII. ACKNOWLEDGMENTS

This work was supported by the European Commission through the EU Projects FEELIX-GROWING (FP6-IST-045169), ROBOT@CWE (FP6-034002), and RobotCub (IST-004370).

The authors warmly thank Auke Jan Ijspeert for support toward the research work developed here.

The authors thank Florent D'Halluin for help in data collection.

References

M. Admiraal and M. Kusters. Modeling kinematics and dynamics of human arm movements. *Motor Control*, 5: 312–338, 2004.

TABLE VI
APPENDIX I. COMPARISON OF THE PROPOSED METHOD WITH [HOFFMANN ET AL., 2009]

The method proposed in this paper:

a single multidimensional system is running to control several DOFs

(A-I-1) $$\dot{\mathbf{x}} = \hat{f}(\mathbf{x})$$
(A-I-2) $$\hat{f}(\mathbf{x}) = \sum_{k=1}^{K} h_k(\mathbf{x})(\mu_{k,\dot{\mathbf{x}}} + \Sigma_{k,\dot{\mathbf{x}}\mathbf{x}}\Sigma_{k,\mathbf{x}}^{-1}(\mathbf{x} - \mu_{k,\mathbf{x}}))$$

where $\mathbf{x} \in \mathbb{R}^N$; $\Sigma_{k,\dot{\mathbf{x}}\mathbf{x}},\ \Sigma_{k,\mathbf{x}} \in \mathbb{R}^{N\times N}$ are estimated matrices
$\mu_{k,\dot{\mathbf{x}}},\ \mu_{k,\mathbf{x}} \in \mathbb{R}^N$ are estimated vectors

Dynamic Movement Primitives proposed in [Hoffmann et al., 2009]:

the acceleration along each DOF $\dot{x}$ is defined by according to:

(A-I-3) $$\tau\ddot{x} = -D\dot{x} + K(g - x) - K(g - x_0)s + K\hat{f}(s)$$

(A-I-4) $$\hat{f}(s) = \frac{\sum_{k=1}^{K} s\Psi_k(s)\omega_k}{\sum_{k=1}^{K}\Psi_k(s)}$$

where $x,\ s, \in \mathbb{R}$
$\Psi_k(s) = \exp\frac{(s-c_k)^2}{2\sigma_k^2}$, $\omega_k \in \mathbb{R}$.

The canonical variable s is governed by a dynamical system:

(A-I-5) $$\tau\dot{s} = -\alpha_s s, \quad s \in [0..1];\ s(0) = 1$$

where $g,\ \alpha_s,\ K,\ D \in \mathbb{R}$ are known constants

Comparison between $\hat{f}(\mathbf{x})$ from our approach and $\hat{f}(s)$ from DMP:

The function $f(\mathbf{x})$ aims at encoding an actual dependency between position and velocity of along a motion; it therefore simultaneously provides the feedback loop for motion adaptation. On the other hand, the function $\hat{f}(s)$ encodes a particular acceleration profile as a function of the internal counter s. The variable s is not linked with positional information and, hence, $\hat{f}(s)$ does not provide the necessary feedback to adapt the motion. Due to such a choice of learning variables, DMP perform the scaling of a demonstrated trajectory, but do not provide the actual robust adaptation to perturbations that can be generated by our system.

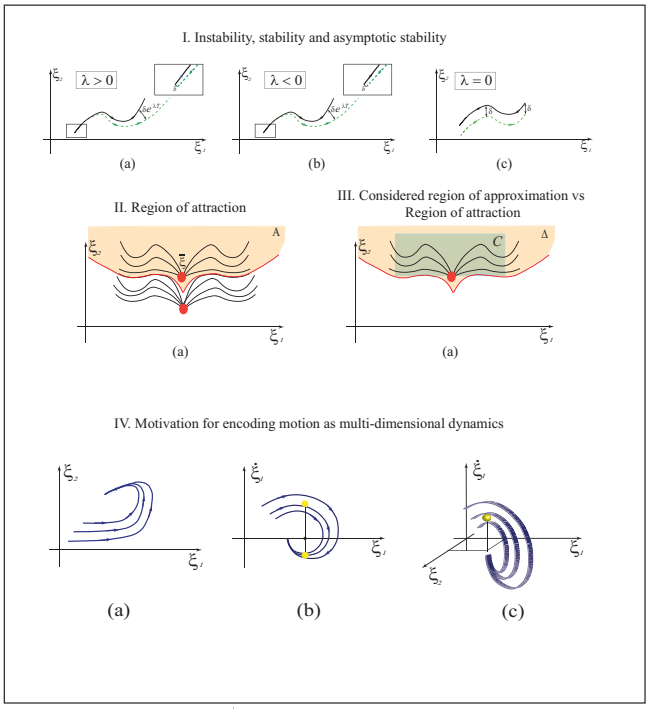

Fig. 30. Appendix II. Geometrical illustration of stability and multi-dimensional correlation in the state-space. I. ***Stability problem***: stability of a dynamical system is defined by a maximum value of its *Lyapunov exponent* λ (in the linear case, it coincides with eigenvalues of a control matrix). (a) In systems with *negative* Lyapunov exponents volume between trajectories contracts; (b) In systems with *positive* Lyapunov exponents two arbitrary near trajectories diverge from each other exponentially fast. In the linear case, one may easily find Lyapunov exponents and estimate the global behavior of the overall system. In the non-linear case, the system may have different Lyapunov exponents in different parts of the state-space, moreover, non-linearities make analytical investigation of properties particularly tedious. IV. ***Multi-dimensional dynamics*** Analyzing dynamics of vector-valued timeseries requires their encoding in multi-dimensional state-spaces. Generally, one cannot unambiguously decouple dynamics of each dimension. Consider a simple 2D motion in Figure II-(a), the phase-space of this motion in $\{\dot{x}_1, x_1\}$ is in Figure II-(b): for each value x_1 there exist two different values of velocity, therefore, it is not possible to unambiguously encode dynamics of motion as two decoupled system $\dot{x}_1 = f_1(x_1)$, $\dot{x}_2 = f_2(x_2)$. However, if one look at the dependency $\dot{x}_1 = f(x_1, x_2)$ depicted at Figure II-(c) this ambiguity can be easily eliminated. This problem is know in the literature on Dynamical Systems as a problem of searching for a *minimum embedding dimension*. In this particular example, the minimum embedding dimension is 4 $(x_1, \dot{x}_1, x_2, \dot{x}_2)$. Alternatively, one may argue that in this case we may avoid an ambiguity and separate dimensions encoding $\ddot{x}_1 = f_1(x_1, \dot{x}_1)$, though it is possible in this particular case, it will lead to the necessity to analyze 5 state variables $(x_1, \dot{x}_1, \ddot{x}_1, x_2, \dot{x}_2)$. Furthermore, to preserve a spatial correlation pattern between x_1 and x_2 the decoupled systems should be synchronized by an external mechanism.

J. Aleotti, S. Caselli, and M. Reggiani. Evaluation of virtual fixtures for a robot programming by demonstration interface. *Transactions on Systems, Man, and Cybernetics*, 35(4): 536–545, 2005.

R. Andersson. Aggressive trajectory generator for a robot ping-pong player. In *IEEE Control Systems Magazine*, volume 9, pages 15–21, Feb 1989.

M. Aoki. *State Space Modeling of Time Series*. Springer-Verlag, 1990.

C. Atkeson and S. S. Moore, A. Locally weighted learning. 1996.

H. Attias. Inferring parameters and structure of latent variable models by variational bayes. In *Uncertainty in Artificial Intelligence*, 1999.

X. Bai, X.-S. Yang, and H. Li. Estimates of the region of attraction of continuous-time cascade systems. *Journal of Mathematical Control and Information*, 24(4):483–491, 2007.

R. Bellman. *Dynamic Programming*. NJ:Princeton Univ. Press, 1957.

B. Berret, C. Darlot, F. Jean, T. Pozzo, C. Papaxanthis, and G. J.-P. The inactivation principle: Mathematical solutions minimizing the absolute work and biological implications for the planning of arm movements. *Journal of Computational Biology*, 4(10), 2008.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot Programming by Demonstration. In *Handbook of Robotics*, volume chapter 59. MIT Press, 2008.

E. Bizzi, N. Accornero, W. Chapple, and N. Hogan. Posture control and trajectory formation during arm movement. *The Journal of Neuroscience*, 4:2738–2744, 1984.

O. Brock, J. Kuffner, and J. Xiao. *Handbook of Robotics*, chapter "Motion for manipulation tasks". Springer-Verlag, 2008.

M. D. Buhmann. *Radial Basis Functions: Theory and Imple-*

mentations. Cambridge University Press, 2003.

D. Bullock and S. Grossberg. Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review*, 95:49–90, 1988.

S. Calinon and A. Billard. What is the Teacher's Role in Robot Programming by Demonstration? - toward Benchmarks for Improved Learning. *Interaction Studies. Special Issue on Psychological Benchmarks in Human-Robot Interaction*, 8 (3), 2007.

S. Calinon and A. Billard. A Probabilistic Programming by Demonstration Framework Handling Constraints in Joint Space and Task Space. In *Proceedings of the International Conference of Intelligent Robots and Systems*, 2008.

S. Calinon, F. Guenter, and A. Billard. On Learning, Representing and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, 37(2):286–298, 2007.

T. L. Carroll. A nonlinear dynamics method for signal identification. *Chaos*, 17, 2007.

F. Chamroukhi, A. Same, G. Govaert, and P. Aknin. Time series modelling by a regression approach based on a latent process. *Neural Networks.*, 22:593–602, 2009.

A. Coates, P. Abbeel, and A. Ng. Learning for control from multiple demonstrations. In *Proceedings of the International Conference on Machine learning*, 2008.

M. B. Crutchfield J.P. Equation of motion from a data series. *Complex Systems*, 1:417–452, 1987.

A. d'Avella and M. Tresch. Modularity in the motor system: decomposition of muscle patterns as combinations of time-varying synergies. In *Advances in Neural Information Processing 14*, 2002.

A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature Neuroscience*, 6:300–308, 2003.

M. Deisenroth, C. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72:1508–1524, 2009.

Y. Demiris and B.Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54:361–369, 2006.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistic Society*, 39:1–38, 1977.

R. Diankov and J. James Kuffner. Randomized statistical path planning. In *Proceedings of IEEE/RSJ International Conference on Robots and Systems (IROS)*, 2007.

K. Dixon and P. Khosla. Trajectory representation using sequenced linear dynamical systems. *Proceedings of International Conference on Robotics and Automation*, 4:3925–3930, 2004.

T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience*, 5(7):1688–1703, 1985.

R. Genesio, M. Tartaglia, and A. Vicino. On the estimation of asympototic stability regions: state of the art and new proposals. *IEEE Transaction on Automatic Control*, 30(8), 1985.

P. Giesl. Construction of a local and global lyapunov function using radial basis functions. *Journal of Applied Mathematics*, 73(5):782–802, 2008.

S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi. Convergent force fields organized in the frogs spinal cord. *The Journal of Neuroscience*, 13(2):467–491, 1993.

E. Gribovskaya and A. Billard. Combining Dynamical Systems Control and Programming by Demonstration for Teaching Discrete Bimanual Coordination Tasks to a Humanoid Robot. In *IEEE/ASM International Conference on Human-Robot Interaction*, 2008.

E. Gribovskaya and A. Billard. Learning Nonlinear Dynamics of Motion for Position and Orientation Control of Robotic Manipulators. In *9th IEEE-RAS International Conference on Humanoid Robots*, 2009.

F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement Learning for Imitating Constrained Reaching Movements. *RSJ Advanced Robotics, Special Issue on Imitative Robots*, 21(13):1521–1544, 2007.

W. Hardle. *Smoothing Techniques with Implementation in Statistics*. NY:Springer, 1991.

C. Harris and D. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780–784, 1998.

M. Hersch and A. Billard. Reaching with multi-referential dynamical systems. *Autonomous Robots*, 25:71–83, 2008.

M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations. *IEEE Transactions on Robotics*, 1, 2008.

P. A. Hinrichsen D. *Mathematical Systems Theory*. Springer Berlin, 2000.

H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In *Proceedings of International Conference on Robotics and Automation*, 2009.

P. Hsu, J. Hauser, and S. Sastry. Dynamic control of redundant manipulators. *Journal of Robotic Systems*, 6(2):133148, 1989.

J.-H. Hwang, R. Arkin, and D.-S. Kwon. Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. volume 2, pages 1444–1449 vol.2, Oct. 2003.

A. Ijspeert and A. Crespi. Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. *Proceedings of International Conference on Robotics and Automation*, 1:262–268, April 2007.

A. Ijspeert, J. Nakanishi, and S. Schaal. Trajectory formation for imitation with nonlinear dynamical systems. *Proceedings of International Conference on Intelligent Robots and Systems*, 2:752–757, 2001.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 958–963, 2002.

T. Kang, J. He, and S. Tillery. Determining natural arm configuration along a reaching trajectory. *Journal of Experimental Brain Research*, pages 352–361, 2005.

J. A. S. Kelso. *Dynamic patterns: The self-organization of brain and behavior*. Cambridge: MIT Press., 1995.

H. Khalil. *Nonlinear systems*. Prentice Hall Upper Saddle River, NJ, 1996.

O. Khatib. A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE Journal of Robotics and Automation*, 3(1):43–53, 1987.

J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

D. Kulic, W. Takano, and Y. Nakamura. Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *The International Journal of Robotics Research*, 27(7):761–784, 2008.

E. T. Y. Lee. Comments on some b-spline algorithms. *Computing (Springer-Verlag)*, 36:229–238, 1986.

L. Ljung. State of the art in linear system identification: Time and frequency domain methods. *Proceeding of the American Control Conference*, 1:650–661, 2004.

M. Loccufier and E. Noldus. A new trajectory reversing method for estimating stability regions of autonomous nonlinear systems. *Nonlinear Dynamics*, 21(3):265–288, 2000.

G. McLahlan and D. Peel. *Finite Mixture Models*. NY:Wiley, 2000.

A. Moore. *Efficient memory-based learning for robot control*. PhD thesis, University of Cambridge, 1990.

H.-G. Muller. *Nonparametric Regression Analysis of Longitudinal Data*. Berlin:Springer, 1988.

J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal. Comparative experiments on task space control with redundancy resolution. In *Proceedings of International Conference on Intelligent Robots and Systems*, 2005.

D. Nguyen-Tuong, M. Seeger, and J. Peters. Local gaussian process regression for real time online model learning and control. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2008.

a. H. H. Park, D., P. Pastor, and S. Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *In the Proceeding of the IEEE International Conference on Humanoid Robotics*, 2008.

J. Park and I. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246 – 257, 1991.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *Proceedings of International conference on Robotics and Automation*, 2009.

J. Peters and S. Schaal. Learning to control in operational space. *International Journal of Robotic Research*, 27(2): 197–212, 2008.

B. Petreska and A. Billard. Movement curvature planning through force field internal models. *Biological Cybernetics*, pages 331–350, 2009.

M. Priestley. State-dependent models: A general approach to non-linear time series analysis. *Journal of Time Series Analysis*, 1, 1980.

L. Righetti, J. Buchli, and A. Ijspeert. Dynamic hebbian learning in adaptive frequency oscillators. *Physica D*, 216 (2):269–281, 2006.

R. W. L. Ryoung K. Lim, Minh Q. Phan. State-space system identification with identified hankel matrix. Technical report, Department of Mechanical and Aerospace Engineering Technical Report No.3045, Princeton University, Princeton, NJ., 1998.

S. Schaal and C. Atkeson. Robot juggling: implementation of memory-based learning. *IEEE Control Systems Magazine*, 14(1):57–71, Feb 1994.

S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

S. Schaal, S. Kotosaka, and D. Sternard. Nonlinear dynamical systems as movement primitives. In *Proceedings of the International Conference on Humanoid Robotics*, 2001.

S. Schaal, A. Ijspeert, and A. Billard. Computational Approaches to Motor Learning by Imitation. *Philosophical transactions: biological sciences*, 358(1431):537–547, 2003.

S. Schaal, P. Mohajerian, and A. Ijspeert. Dynamics systems vs. optimal control a unifying view. *Progress in Brain Research*, 165:425–445, 2007.

G. Schoner and C. Santos. Control of movement time and sequential action through attractor dynamics: A simulation study demonstration object perception and coordination. In *Symposium on Intelligent Robotic Systems*, 2001.

R. Shadmehr, M. Smith, and W. Krakauer. Error correction. sensory prediction, and adaptation in mirror control. *Annual Reviews of Neuroscience*, 2010.

D. Sternad and D. Schaal. Segmentation of endpoint trajectories does not imply segmented control. *Experimental Brain Research*, 124:118–136, 1999.

H. G. Sung. *Gaussian Mixture Regression*. PhD thesis, Rice Univercity, Huston, Texas, 2004.

R. Tedrake, I. Manchester, M. Tobenkin, and J. Roberts. Lqr-trees: Feedback motion planning via sums of squares verification. *International Journal of Robotics Research*, 2010.

E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11): 1226–1235, 2002.

H. Tomohisa, W. M. Haddad, and H. Naira. Neural network adaptive control for a class of nonlinear uncertain dynamical systems with asymptotic stability guarantees. *IEEE Transactions on Neural Networks*, 19:80–90, 2008.

D. Travis, B. T. Thumati, and S. Jagannathan. Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence. *Neural Networks*, 22:851–860, 2009.

A. Ude, C. Atkeson, and M. Riley. Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems*, 47:93–108, 2004.

S. Vijayakumar and S. Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *Proceedings of the International Conference on Machine Learning*, pages 1079–1086, 2000.

P. Vivani and C. Terzuolo. Trajectory determines movement

dynamics. *Neuroscience*, 7:431–437, 1982.

S. Wang, H. Luo, C. Yue, and X. Liao. Parameter identification of chaos system based on unknown parameter observer. *Physics letters. A*, 372:2603–2607, 2008.

H. Wei and S. Amari. Dynamics of learning near singularities in radial basis function networks. *Neural Networks*, 21: 981–1005, 2008.

N. Xie and H. Leung. Blind identification of autoregressive system using chaos. *IEEE Transactions on Circuits and Systems*, 52:1953–1965, 2005.

K. Yamane, J. Kuffner, and J. Hodgins. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics*, 23(3):532–539, 2004.

K. Yokoi, E. Yoshida, and H. Sanada. Unified motion planning of passing under obstacles with humanoid robots. In *Proceedings of International Conference on Robotics and Automation*, pages 1185–1190, May 2009.

E. Yoshida, M. Poirier, J. Laumond, O. Kanoun, F. Lamiraux, R. Alami, and K. Yokoi. Whole-body motion planning for pivoting based manipulation by humanoids. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2008.

R. Zollner, T. Afour, and D. R. Programming by demonstration: Dual-arm manipulation tasks for humanoid robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2004.