Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models

S. Mohammad Khansari-Zadeh and Aude Billard

***Abstract*—This paper presents a method to learn discrete robot motions from a set of demonstrations. We model a motion as a nonlinear autonomous (i.e., time-invariant) dynamical system (DS) and define sufficient conditions to ensure global asymptotic stability at the target. We propose a learning method, which is called *Stable Estimator of Dynamical Systems (SEDS)*, to learn the parameters of the DS to ensure that all motions closely follow the demonstrations while ultimately reaching and stopping at the target. Time-invariance and global asymptotic stability at the target ensures that the system can respond immediately and appropriately to perturbations that are encountered during the motion. The method is evaluated through a set of robot experiments and on a library of human handwriting motions.**

***Index Terms*—Dynamical systems (DS), Gaussian mixture model, imitation learning, point-to-point motions, stability analysis.**

I. Introduction

We consider modeling of point-to-point motions, i.e., movements in space stopping at a given target [1]. Modeling point-to-point motions provides basic components for robot control, whereby more complex tasks can be decomposed into sets of point-to-point motions [1], [2]. As an example, consider the standard "pick-and-place" task: First, reach for the item, then after grasping, move to the target location, and finally, return home after release.

Programming by demonstration (PbD) is a powerful means to bootstrap robot learning by providing a few examples of the task at hand [1], [3]. We consider PbD of point-to-point motions where motions are performed by a human demonstrator. To avoid addressing the correspondence problem [4], motions are demonstrated from the robot's point of view by the user that guides the robot's arm passively through the task. In our experiments, this is done either by back driving the robot or by teleoperating it using motion sensors (see Fig. 1). We, hence, focus on the "what to imitate" problem [4] and derive a means to extract the generic characteristics of the dynamics of the motion. In this paper, we assume that the relevant features of the movement, i.e., those to imitate, are the features that appear most frequently, i.e., the invariants across the demonstration. As a result, demonstrations should be such that they contain the main features of the desired task, while exploring some of the variations allowed within a neighborhood around the space covered by the demonstrations.

Fig. 1. Demonstrating motions by teleoperating a robot (left) using motion sensors or (right) by back driving it.

A. Formalism

We formulate the encoding of point-to-point motions as control law that is driven by autonomous dynamical systems (DS): Consider a state variable $\xi \in \mathbb{R}^d$ that can be used to *unambiguously* define a discrete motion of a robotic system (e.g., ξ could be a robot's joint angles, the position of an arm's end-effector in the Cartesian space, etc.). Let the set of N given demonstrations $\{\xi^{t,n}, \dot{\xi}^{t,n}\}_{t=0,n=1}^{T^n,N}$ be instances of a global motion model that is governed by a first-order autonomous ordinary differential equation (ODE)

$$\dot{\xi} = f(\xi) + \epsilon \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is a nonlinear continuous and continuously differentiable function with a single equilibrium point $\dot{\xi}^* = f(\xi^*) = 0$, $\boldsymbol{\theta}$ is the set of parameters of f, and ϵ represents a zero mean additive Gaussian noise. The noise term ϵ encapsulates both inaccuracies in sensor measurements and errors that result from imperfect demonstrations. The function $\hat{f}(\xi)$ can be described by a set of parameters $\boldsymbol{\theta}$, in which the optimal values of $\boldsymbol{\theta}$ can be obtained based on the set of demonstrations using different statistical approaches.[1] We will further denote the obtained noise-free estimate of f from the statistical modeling with $\hat{f}$ throughout this paper. Our noise-free estimate will, thus, be

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

[1] Assuming a zero mean distribution for the noise makes it possible to estimate the noise free model through regression.

Manuscript received June 24, 2010; revised December 16, 2010 and April 27, 2011; accepted June 3, 2011. Date of publication July 14, 2011; date of current version October 6, 2011. This paper was recommended for publication by Associate Editor S. Hutchinson and Editor K. Lynch upon evaluation of the reviewers' comments. This work was supported by the European Commission through the European Union Project Adaptive Modular Architectures for Rich Motor Skills under Contract FP7-ICT-248311.

The authors are with the School of Engineering, Ecole Polytechnique Federale de Lausanne, Lausanne 1015, Switzerland (e-mail: mohammad.khansari@epfl.ch; aude.billard@epfl.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2011.2159412

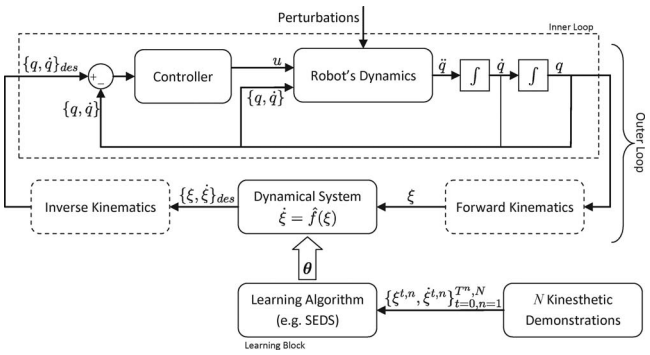

Fig. 2. Typical system's architecture that illustrates the control flow in a robotic system as considered in this paper. The system is composed of two loops: the inner loop that represents the robot's dynamics and a low-level controller and an outer loop that defines the desired motion at each time step. The learning block is used to infer the parameters of motion $\boldsymbol{\theta}$ from demonstrations.

Given an arbitrary starting point $\xi^0 \in \mathbb{R}^d$, the evolution of motion can be computed by integrating from (2).

Two observations follow from formalizing our problem using (1) and (2): 1) The control law that is given by (2) will generate trajectories that do not intersect, even if the original demonstrations did intersect; and 2) the motion of the system is uniquely determined by its state ξ. The choice of state variable ξ is, hence, crucial. For instance, if one wishes to represent trajectories that intersect in the state space, one should encode both velocity and acceleration in ξ, i.e., $\xi = [x; \dot{x}]$.

The use of DS is advantageous in that it enables a robot to adapt its trajectory *instantly* in the face of perturbations [5]. A controller that is driven by a DS is robust to perturbations because it embeds all possible solutions to reach a target into one single function $\hat{f}$. Such a function represents a global map that specifies on the fly the correct direction for reaching the target, considering the current position of the robot and the target. In this paper, we consider two types of perturbations: 1) *spatial perturbations* that result from a sudden displacement in space of either the robot's arm or of the target; and 2) *temporal perturbations* which result from delays in the execution of the task.[2]

Throughout this paper, we choose to represent a motion in a kinematic coordinates system (i.e., the Cartesian or robot's joint space) and assume that there exists a low-level controller that converts kinematic variables into motor commands (e.g., force or torque). Fig. 2 shows a schematic of the control flow. The whole system's architecture can be decomposed into two loops. The inner loop consists of a controller that generates the required commands to follow the desired motion and a system block to model the dynamics of the robot. Here, q, $\dot{q}$, and $\ddot{q}$ are the robot's joint angle and its first and second time derivatives. Motor commands are denoted by u. The outer loop specifies the next desired position and velocity of the motion with respect to the current status of the robot. An inverse kinematics block may also be considered in the outer loop to transfer the desired trajectory from the Cartesian to the joint space (this block is not necessary if the motion is already specified in the joint space).

In this control architecture, both the inner and outer loops should be stable. The stability of the inner loop requires the system to be input-to-state stable (ISS) [6], i.e., the output of the inner loop should remain bounded for a bounded input. The stability of the outer loop is ensured when learning the system. The learning block refers to the procedure that determines a stable estimate of the DS to be used as the outer-loop control. In this paper, we assume that there exists a low-level controller which is not necessarily accurate,[3] that makes the inner-loop ISS. Hence, we focus our efforts on designing a learning block that ensures stability of the outer-loop controller. Learning is data driven and uses a set of demonstrated trajectories to determine the parameters $\boldsymbol{\theta}$ of the DS that is given in (2). Learning proceeds as a constraint optimization problem, satisfying asymptotic stability of the DS at the target. A formal definition of stability is given next.

Definition 1: The function $\hat{f}$ is globally asymptotically stable at the target ξ^* if $f(\xi^*) = 0$ and $\forall \xi^0 \in \mathbb{R}^d$; the generated motion converges asymptotically to ξ^*, i.e.,

$$\lim_{t\to\infty} \xi^t = \xi^* \qquad \forall \xi^0 \in \mathbb{R}^d. \tag{3}$$

$\hat{f}$ is locally asymptotically stable if it converges to ξ^* only when ξ^0 is contained within a subspace $D \subset \mathbb{R}^d$.

Nonlinear DS are prone to instabilities. Ensuring that the estimate $\hat{f}$ results in *asymptotically stable trajectories*, i.e., trajectories that converge asymptotically to the attractor as per Definition 1, is thus a key requirement for $\hat{f}$ to provide a useful control policy. In this paper, we formulate the problem to estimate f and its parameters $\boldsymbol{\theta}$ as a constrained optimization problem, whereby we maximize accuracy of the reconstruction while ensuring its global asymptotic stability at the target.

The remainder of this paper is structured as follows. Section II reviews related works on learning discrete motions and the shortcomings of the existing methods. Section III formalizes the control law as a stochastic system composed of a mixture of Gaussian functions. In Section IV, we develop conditions to ensure global asymptotic stability of nonlinear DS. In Section V, we propose a learning method to build an ODE model that satisfies these conditions. In Section VI, we quantify the performance of our method to estimate the dynamics of motions 1) against a library of human handwriting motions; and 2) in two different robot platforms (i.e., the humanoid robot iCub and the industrial robot Katana-T). We further demonstrate how the resulting model from the proposed learning methods can adapt instantly to temporal and spatial perturbations. We devote Section VII to discussion, and finally, we summarize the obtained results in Section VIII.

[2]Note that we distinguish between spatial and temporal perturbations as these result in different distortion of the estimated dynamics and, hence, require different means to tackle these. Typically, spatial perturbations would result from an imprecise localization of the target or from interacting with a dynamic environment where either the target or the robot's arm may be moved by an external perturbation; temporal perturbations typically arise when the robot is stopped momentarily due to the presence of an object or due to safety issues (e.g., waiting until the operator has cleared the workspace).

[3]When controlled by a DS, the outer-loop controller can handle the inner-loop controller's inaccuracy by treating these as perturbations, comparing the expected versus the actual state of the system.

II. Related Works

Statistical approaches to modeling robot motion have become increasingly popular as a means to deal with the noise inherent in any mechanical system. They have proved to be interesting alternatives to classical control and planning approaches when the underlying model cannot be well estimated. Traditional means of encoding trajectories is based on spline decomposition after averaging across training trajectories [7]–[10]. While this method is a useful tool for quick and efficient decomposition and generalization over a given set of trajectories, it is, however, heavily dependent on heuristics to segment and align the trajectories and gives a poor estimate of nonlinear trajectories.

Some alternatives to spline-based techniques perform regression over a nonlinear estimate of the motion that is based on Gaussian kernels [2], [11], [12]. These methods provide powerful means to encode arbitrary multidimensional nonlinear trajectories. However, similar to spline encoding, these approaches depend on explicit time indexing and virtually operate in an open loop. Time dependence makes these techniques very sensitive to both temporal and spatial perturbations. To compensate for this deficiency,[4] one requires a heuristic to reindex the new trajectory in time, while simultaneously optimizing a measure of how good the new trajectory follows the desired one. To find a good heuristic is highly task-dependent and a nontrivial task, and becomes particularly nonintuitive in high-dimensional state spaces.

Coates *et al.* [13] proposed an Expectation Maximization (EM) algorithm that uses an (extended) Kalman smoother to follow a desired trajectory from the demonstrations. They use dynamic programming to infer the desired target trajectory and a time alignment of all demonstrations. Their algorithm also learns a local model of the robot's dynamics along the desired trajectory. Although this algorithm is shown to be an efficient method to learn complex motions, it is time dependent and, thus, shares the disadvantages that are mentioned earlier.

DS have been advocated as a powerful alternative to modeling robot motions [5], [14]. Existing approaches to the *statistical* estimation of f in (2) use either Gaussian Process Regression (GPR) [15], Locally Weighted Projection Regression (LWPR) [16], or Gaussian Mixture Regression (GMR) [14], where the parameters of the Gaussian Mixture are optimized through EM [17]. GMR and GPR find a locally optimal model of $\hat{f}$ by maximizing the likelihood that the complete model represents the data well, while LWPR minimizes the mean square error (MSE) between the estimates and the data (for a detailed discussion on these methods, see [18]).

Because all of the aforementioned methods do not optimize under the constraint of making the system stable at the attractor, they are not guaranteed to result in a stable estimate of the motion. In practice, they fail to ensure global stability, and they also rarely ensure local stability of $\hat{f}$ (see Definition 1). Such estimates of the motion may, hence, converge to spurious attractors or miss the target (diverging/unstable behavior) even when estimating simple motions such as motions in the plane, see Fig. 3. This is due to the fact that there is yet no generic theoretical solution to ensuring stability of arbitrary nonlinear autonomous DS [19]. Fig. 3 illustrates an example of unstable estimation of a nonlinear DS using the aforementioned three methods for learning a 2-D motion. Fig. 3(a) represents the stability analysis of the dynamics learned with GMR. Here, in the narrow regions around demonstrations, the trajectories converge to a spurious attractor just next to the target. In other parts of the space, they either converge to other spurious attractors far from the target or completely diverge from it. Fig. 3(b) shows the obtained results from LWPR. All trajectories inside the black boundaries converge to a spurious attractor. Outside of these boundaries, the velocity is always zero (a region of spurious attractors); hence, a motion stops once it crosses these boundaries or it does not move when it initializes there. Regarding Fig. 3(c), while for GPR trajectories converge to the target in a narrow area close to demonstrations, they are attracted to spurious attractors outside that region.

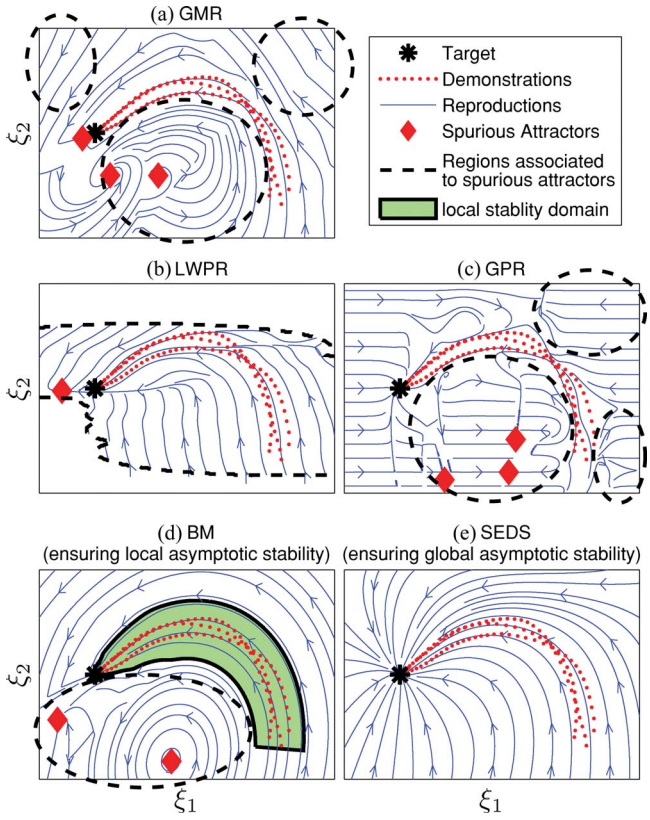

Fig. 3. Example of 2-D dynamics learned from three demonstrations using five different methods: GMR, LWPR, GPR, BM, and SEDS (this study). For further information, see the text.

In all these examples, regions of attractions are usually very close to demonstrations and, thus, should be carefully avoided. However, the critical concern is that there is no generic theoretical solution to determine beforehand whether a trajectory will lead to a spurious attractor, to infinity, or to the desired attractor. Thus, it is necessary to conduct numerical stability analysis to

[4]If one is to model only time-dependent motions, i.e., motions that are deemed to be performed in a fixed amount of time, then one may prefer a time-dependent encoding.

locate the region of attraction of the desired target which may never exist or may be very narrow.

The Dynamic Movement Primitives (DMP) [20] offer a method by which a nonlinear DS can be estimated while ensuring global stability at an attractor point. Global stability is ensured through the use of linear DS that takes precedence over the nonlinear modulation to ensure stability at the end of the motion. The switch from nonlinear to linear dynamics proceeds smoothly according to a phase variable that acts as an implicit clock. Such an implicit time dependence requires a heuristics to reset the phase variable in the face of temporal perturbations. When learning from a single demonstration, DMP offers a robust and precise means of encoding a complex dynamics. Here, we take a different approach in which we aim at learning a generalized dynamics from *multiple* demonstrations. We also aim to ensure time independence, and hence robustness to temporal perturbations. Learning also proceeds from extracting correlation across several dimensions. While DMP learns a model for each dimension separately, we here model a single multidimensional model. The approach that we propose is, hence, complementary to DMP. The choice between using DMP or stable estimator of dynamical systems (SEDS) to model a motion is application dependent. For example, when the motion is intrinsically time dependent and only a single demonstration is available, one may use DMP to model the motion. In contrast, when the motion is time independent and when learning from multiple demonstrations, one may opt to use SEDS. For a more detailed discussion of these issues and for quantitative comparisons across time-dependent and time-independent encoding of motions using DS, see [5] and [21].

In our prior work [14], we developed a hybrid controller that is composed of two DS working concurrently in end-effector and joint angle spaces, resulting in a controller that has no singularities. While this approach was able to adapt online to sudden displacements of the target or unexpected movement of the arm during the motion, the model remained time dependent because, similarly to DMP, it relied on a stable linear DS with a fixed internal clock.

We, then, considered an alternative DS approach that is based on the hidden Markov model and GMR [22]. The method that is presented here is time independent and, thus, robust to temporal perturbations. Asymptotic stability could, however, not be ensured. Sole a brief verification to avoid large instabilities was done by evaluating the eigenvalues of each linear DS and ensuring that they all have negative real parts. As stated in [22] and as we will show in Section IV, asking that all eigenvalues be negative is not a sufficient condition to ensure stability of the complete system (see, e.g., Fig. 5).

In [21] and [23], we proposed a heuristics to build iteratively a *locally* stable estimate of nonlinear DS. This heuristics requires one to increase the number of Gaussians and retrain the mixture using EM iteratively until stability can be ensured. Stability was tested *numerically*. This approach suffered from the fact that it was not ensured to find a (*even locally*) stable estimate and that it gave no explicit constraint on the form of the Gaussians to ensure stability. The model had a limited domain of applicability because of its local stability, and it was also computationally intensive, making it difficult to apply the method in high dimensions.

In [18], we proposed an iterative method, which is called Binary Merging (BM), to construct a mixture of Gaussians so as to ensure *local* asymptotic stability at the target; hence, the model can be only applied in a region close to demonstrations [see Fig. 3(d)]. Although this study provided sufficient conditions to make DS *locally* stable, similar to [23], it still relied on determining numerically the stability region and had a limited region of applicability.

In this paper, we develop a *formal* analysis of stability and formulate explicit constraints on the parameters of the mixture to ensure *global* asymptotic stability of DS. This approach provides a sound ground for the estimation of nonlinear DS which is not heuristic driven and, thus, has the potential for much larger sets of applications, such as the estimation of second-order dynamics and for control of multidegrees of freedom (multi-DOF) robots as we demonstrate here. Fig. 3(e) represents results that are obtained in this paper. Being globally asymptotically stable, all trajectories converge to the target. This ensures that the task can be successfully accomplished starting from any point in the operational space with no need to reindex or rescale. Note that the stability analysis that we presented here was published in a preliminary form in [5]. This paper largely extends this work by 1) having a more depth discussion on stability; 2) by proposing two objective functions to learn parameters of DS and comparing their pros and cons; 3) by having a more detailed comparison of the performance of the proposed method with BM and three best regression methods to estimate motion dynamics, namely GMR, LWPR, and GPR; and 4) by having more robot experiments.

III. Multivariate Regression

We use a probabilistic framework and model $\hat{f}$ via a finite mixture of Gaussian functions. Mixture modeling is a popular approach for density approximation [24], and it allows a user to define an appropriate model through a tradeoff between model complexity and variations of the available training data. Mixture modeling is a method that builds a coarse representation of the data density through a fixed number (usually lower than 10) of mixture components. An optimal number of components can be found using various methods, such as the Bayesian information criterion (BIC) [25], the Akaike information criterion (AIC) [26], the deviance information criterion (DIC) [27], that penalize a large increase in the number of parameters when it only offers a small gain in the likelihood of the model.

While nonparametric methods, such as Gaussian Process or variants on these, offer optimal regression [15], [28], they suffer from the curse of dimensionality. Indeed, computing the estimate regressor $\hat{f}$ grows linearly with the number of data points, making such an estimation inadequate for on-the-fly recomputation of the trajectory in the face of perturbations. There exists various sparse techniques to reduce the sensitivity of these methods to the number of data points. However, these techniques either become parametric by predetermining the optimal number of data points [29], or they rely on a heuristic such as

information gain to determine the optimal subset of data points [30]. These heuristics resemble that offered by the BIC, DIC, or AIC criteria.

Estimating f via a finite mixture of Gaussian functions, the unknown parameters of $\hat{f}$ become the prior π^k, the mean μ^k and the covariance matrices Σ^k of the $k = 1 \dots K$ Gaussian functions (i.e., $\theta^k = \{\pi^k, \mu^k, \Sigma^k\}$ and $\boldsymbol{\theta} = \{\theta^1 \dots \theta^K\}$). The mean and the covariance matrices of a Gaussian k are defined by

$$\mu^k = \begin{pmatrix} \mu^k_{\xi} \\ \mu^k_{\dot{\xi}} \end{pmatrix}, \quad \Sigma^k = \begin{pmatrix} \Sigma^k_{\xi} & \Sigma^k_{\xi\dot{\xi}} \\ \Sigma^k_{\dot{\xi}\xi} & \Sigma^k_{\dot{\xi}} \end{pmatrix}. \tag{4}$$

Given a set of N demonstrations $\{\xi^{t,n}, \dot{\xi}^{t,n}\}_{t=0,n=1}^{T^n,N}$, each recorded point in the trajectories $[\xi^{t,n}, \dot{\xi}^{t,n}]$ is associated with a probability density function $\mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n})$:

$$\mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n}; \boldsymbol{\theta}) = \sum_{k=1}^{K} \mathcal{P}(k)\mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n}|k) \quad \begin{cases} \forall n \in 1 \dots N \\ t \in 0 \dots T^n \end{cases} \tag{5}$$

where $\mathcal{P}(k) = \pi^k$ is the prior, and $\mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n}|k)$ is the conditional probability density function that is given by

$$\begin{aligned} \mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n}|k) &= \mathcal{N}(\xi^{t,n}, \dot{\xi}^{t,n}; \mu^k, \Sigma^k) \\ &= \frac{1}{\sqrt{(2\pi)^{2d}|\Sigma^k|}} e^{-\frac{1}{2}([\xi^{t,n}, \dot{\xi}^{t,n}]-\mu^k)^T (\Sigma^k)^{-1} ([\xi^{t,n}, \dot{\xi}^{t,n}]-\mu^k)}. \end{aligned} \tag{6}$$

Taking the posterior mean estimate of $\mathcal{P}(\dot{\xi}|\xi)$ yields (as described in [31])

$$\dot{\xi} = \sum_{k=1}^{K} \frac{\mathcal{P}(k)\mathcal{P}(\xi|k)}{\sum_{i=1}^{K} \mathcal{P}(i)\mathcal{P}(\xi|i)} \left(\mu^k_{\dot{\xi}} + \Sigma^k_{\dot{\xi}\xi} \left(\Sigma^k_{\xi}\right)^{-1} \left(\xi - \mu^k_{\xi}\right)\right). \tag{7}$$

The notation of (7) can be simplified through a change of variable. Let us define

$$\begin{cases} A^k = \Sigma^k_{\dot{\xi}\xi} (\Sigma^k_{\xi})^{-1} \\ b^k = \mu^k_{\dot{\xi}} - A^k \mu^k_{\xi} \\ h^k(\xi) = \frac{\mathcal{P}(k)\mathcal{P}(\xi|k)}{\sum_{i=1}^{K} \mathcal{P}(i)\mathcal{P}(\xi|i)}. \end{cases} \tag{8}$$

The substitution of (8) into (7) yields

$$\dot{\xi} = \hat{f}(\xi) = \sum_{k=1}^{K} h^k(\xi)(A^k\xi + b^k). \tag{9}$$

First observe that $\hat{f}$ is now expressed as a nonlinear sum of linear DS. Fig. 4 illustrates the parameters of (8) and their effects on (9) for a 1-D model constructed with three Gaussians. Here, each linear dynamics $A^k\xi + b^k$ corresponds to a line that passes through the centers μ^k with slope A^k. The nonlinear weighting terms $h^k(\xi)$ in (9), where $0 < h^k(\xi) \leq 1$, give a measure of the relative influence of each Gaussian locally. Observe that due to the nonlinear weighting terms $h^k(\xi)$, the resulting function $\hat{f}(\xi)$ is nonlinear and flexible enough to model a wide variety of motions. If one estimates this mixture using classical methods such as EM, one cannot guarantee that the system will be asymptotically stable. The resulting nonlinear model $\hat{f}(\xi)$ usually contains several spurious attractors or limit cycles even for a simple 2-D model (see Fig. 3). Next, we determine sufficient conditions on the learning parameters $\boldsymbol{\theta}$ to ensure asymptotic stability of $\hat{f}(\xi)$.

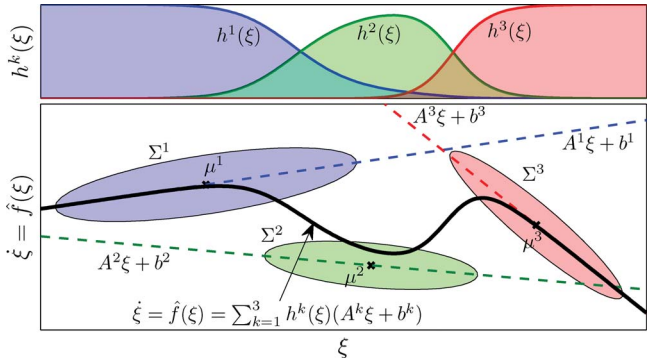

Fig. 4. Parameters that are defined in (8) and their effects on $\hat{f}(\xi)$ for a 1-D model constructed with three Gaussians. See the text for further information.

IV. Stability Analysis

The stability analysis of DS is a broad subject in the field of dynamics and control, which can generally be divided into linear and nonlinear systems. Stability of linear dynamics has been studied extensively [19], where a linear DS can be written as

$$\dot{\xi} = A\xi + b. \tag{10}$$

Asymptotic stability of a linear DS that is defined by (10) can be ensured by solely requiring that the eigenvalues of the matrix A be negative. In contrast, the stability analysis of nonlinear DS is still an open question, and theoretical solutions exist only for particular cases. Beware that the intuition that the nonlinear function $\hat{f}(\xi)$ should be stable if all eigenvalues of matrices A^k, $k = 1 \dots K$, have strictly negative real parts is not true. Here is a simple example in 2-D that illustrates why this is not the case, as well as why estimating stability of nonlinear DS, even in 2-D is nontrivial.

Example: Consider the parameters of a model with two Gaussian functions to be

$$\begin{cases} \Sigma^1_{\xi} = \Sigma^2_{\xi} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \\ \Sigma^1_{\dot{\xi}\xi} = \begin{bmatrix} -3 & -30 \\ 3 & -3 \end{bmatrix}, \quad \Sigma^2_{\dot{\xi}\xi} = \begin{bmatrix} -3 & 3 \\ -30 & -3 \end{bmatrix} \\ \mu^1_{\xi} = \mu^2_{\xi} = \mu^1_{\dot{\xi}} = \mu^2_{\dot{\xi}} = \mathbf{0}. \end{cases} \tag{11}$$

Using (8), we have

$$\begin{cases} A^1 = \begin{bmatrix} -1 & -10 \\ 1 & -1 \end{bmatrix}, \; A^2 = \begin{bmatrix} -1 & 1 \\ -10 & -1 \end{bmatrix} \\ b^1 = b^2 = \mathbf{0}. \end{cases} \tag{12}$$

The eigenvalues of the two matrices A^1 and A^2 are complex with values $-1 \pm 3.16i$. Hence, each matrix determines a stable

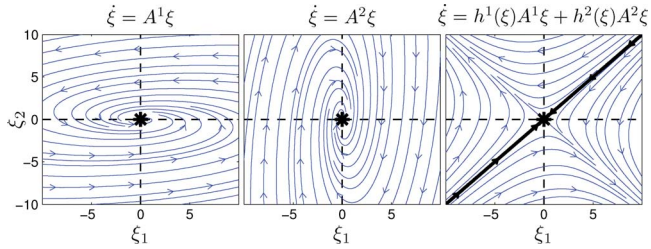

Fig. 5. While each of the subsystems (left) $\dot{\xi} = A^1\xi$ and (center) $\dot{\xi} = A^2\xi$ is asymptotically stable at the origin, a nonlinear weighted sum of these systems (right) $\dot{\xi} = h^1(\xi)A^1\xi + h^2(\xi)A^2\xi$ may become unstable . Here, the system remains stable only for points on the line $\xi_2 = \xi_1$ (drawn in black).

system. However, the nonlinear combination of the two matrices, as per (9), is stable only when $\xi_2 = \xi_1$ and is unstable in $\mathbb{R}^d \setminus \{(\xi_2, \xi_1)|\xi_2 = \xi_1\}$ (see Fig. 5).

Next, we determine sufficient conditions to ensure global asymptotic stability of a series of nonlinear DS given by (7).

Theorem 1: Assume that the state trajectory evolves according to (9). Then, the function that is described by (9) is globally asymptotically stable at the target ξ^* in $\mathbb{R}^d$ if

$$\begin{cases} (\boldsymbol{a})\ b^k = -A^k\xi^* \\ (\boldsymbol{b})\ A^k + (A^k)^T \prec 0 \end{cases} \qquad \forall k = 1 \dots K \tag{13}$$

where $(A^k)^T$ is the transpose of A^k, and $. \prec 0$ refers to the negative definiteness of a matrix.[5]

Proof: We start the proof by recalling the Lyapunov conditions for asymptotic stability of an arbitrary dynamical system [19].

Lyapunov Stability Theorem: A dynamical system that is determined by the function $\dot{\xi} = \hat{f}(\xi)$ is globally asymptotically stable at the point ξ^* if there exists a continuous and continuously differentiable Lyapunov function $V(\xi) : \mathbb{R}^d \to \mathbb{R}$ such that

$$\begin{cases} (\mathbf{a})\ V(\xi) > 0 & \forall \xi \in \mathbb{R}^d, \quad \xi \neq \xi^* \\ (\mathbf{b})\ \dot{V}(\xi) < 0 & \forall \xi \in \mathbb{R}^d, \quad \xi \neq \xi^* \\ (\mathbf{c})\ V(\xi^*) = 0, & \dot{V}(\xi^*) = 0. \end{cases} \tag{14}$$

Note that $\dot{V}$ is a function of both ξ and $\dot{\xi}$. However, since $\dot{\xi}$ can be directly expressed in terms of ξ using (9), one can finally infer that $\dot{V}$ only depends on ξ.

Consider a Lyapunov function $V(\xi)$ of the form

$$V(\xi) = \frac{1}{2}(\xi - \xi^*)^T(\xi - \xi^*) \qquad \forall \xi \in \mathbb{R}^d. \tag{15}$$

Observe first that $V(\xi)$ is a quadratic function and, hence, satisfies condition (14a). Condition that is given by (14b) follows from taking the first derivative of $V(\xi)$ with respect to time; we have

$$\begin{aligned} \dot{V}(\xi) &= \frac{dV}{dt} = \frac{dV}{d\xi}\frac{d\xi}{dt} \\ &= \frac{1}{2}\frac{d}{d\xi}\left((\xi - \xi^*)^T(\xi - \xi^*)\right)\dot{\xi} \\ &= (\xi - \xi^*)^T\dot{\xi} = (\xi - \xi^*)^T\hat{f}(\xi) \\ &= (\xi - \xi^*)^T \underbrace{\sum_{k=1}^{K} h^k(\xi)(A^k\xi + b^k)}_{=\dot{\xi}\ (\text{see}(9))} \\ &= (\xi - \xi^*)^T \sum_{k=1}^{K} h^k(\xi)(A^k(\xi - \xi^*) + \underbrace{A^k\xi^* + b^k}_{=0\ (\text{see}(13a))}) \\ &= (\xi - \xi^*)^T \sum_{k=1}^{K} h^k(\xi)A^k(\xi - \xi^*) \\ &= \sum_{k=1}^{K} \underbrace{h^k(\xi)}_{h^k>0} \underbrace{(\xi - \xi^*)^T A^k(\xi - \xi^*)}_{<0\ (\text{see}(13b))} \\ &< 0 \quad \forall \xi \in \mathbb{R}^d, \quad \xi \neq \xi^*. \end{aligned} \tag{16}$$

Conditions that are given by (14c) are satisfied when substituting $\xi = \xi^*$ into (15) and (16)

$$V(\xi^*) = \frac{1}{2}(\xi - \xi^*)^T(\xi - \xi^*)\Big|_{\xi=\xi^*} = 0 \tag{17}$$

$$\dot{V}(\xi^*) = \sum_{k=1}^{K} h^k(\xi)(\xi - \xi^*)^T A^k(\xi - \xi^*)\Big|_{\xi=\xi^*} = 0. \tag{18}$$

Therefore, an arbitrary ODE function $\dot{\xi} = \hat{f}(\xi)$ that is given by (9) is globally asymptotically stable if conditions of (13) are satisfied. ■

Conditions (13a) and (13b) are sufficient to ensure that an arbitrary nonlinear function that is given by (9) is globally asymptotically stable at the target ξ^*. Such a model is advantageous in that it ensures that starting from any point in the space, the trajectory (e.g., a robot arm's end effector) always converges to the target.

V. Learning Globally Asymptotically Stable Models

Section IV provided us with sufficient conditions whereby the estimate $\hat{f}(\xi)$ is globally asymptotically stable at the target. It remains now to determine a procedure to compute unknown parameters of (9), i.e., $\boldsymbol{\theta} = \{\pi^1 \dots \pi^K; \mu^1 \dots \mu^K; \Sigma^1 \dots \Sigma^K\}$ such that the resulting model is globally asymptotically stable. In this section, we propose a learning algorithm, which is called *SEDS*, that computes optimal values of $\boldsymbol{\theta}$ by solving an optimization problem under the constraint of ensuring the model's global asymptotic stability. We consider two different candidates for the optimization objective function: 1) log-likelihood and 2) MSE. The results from both approaches will be evaluated and compared in Section VI-A.

[5] A $d \times d$ real symmetric matrix A is positive definite if $\xi^T A\xi > 0$ for all nonzero vectors $\xi \in \mathbb{R}^d$, where ξ^T denotes the transpose of ξ. Conversely, A is negative definite if $\xi^T A\xi < 0$. For a nonsymmetric matrix, A is positive (negative) definite if and only if its symmetric part $\tilde{A} = (A + A^T)/2$ is positive (negative) definite.

SEDS-Likelihood: Using log-likelihood as a means to construct a model

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\frac{1}{\mathcal{T}} \sum_{n=1}^{N} \sum_{t=0}^{T^n} \log \mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n} | \boldsymbol{\theta}) \tag{19}$$

subject to

$$\begin{cases} (\boldsymbol{a})\ b^k = -A^k \xi^* \\ (\boldsymbol{b})\ A^k + (A^k)^T \prec 0 \\ (\boldsymbol{c})\ \Sigma^k \succ 0 \\ (\boldsymbol{d})\ 0 < \pi^k \leq 1 \\ (\boldsymbol{e})\ \sum_{k=1}^{K} \pi^k = 1 \end{cases} \quad \forall k \in 1 \ldots K \tag{20}$$

where $\mathcal{P}(\xi^{t,n}, \dot{\xi}^{t,n} | \boldsymbol{\theta})$ is given by (5), and $\mathcal{T} = \sum_{n=1}^{N} T^n$ is the total number of training data points. The first two constraints in (20) are stability conditions from Section IV. The last three constraints are imposed by the nature of the Gaussian mixture model to ensure that Σ^k are positive-definite matrices, priors π^k are positive scalars smaller than or equal to one, and sum of all priors is equal to one (because the probability value of (5) should not exceed 1).

SEDS-MSE: Using MSE as a means to quantify the accuracy of estimations that are based on demonstrations[6]

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{2\mathcal{T}} \sum_{n=1}^{N} \sum_{t=0}^{T^n} \| \hat{\dot{\xi}}^{t,n} - \dot{\xi}^{t,n} \|^2 \tag{21}$$

subject to the same constraints as given by (20). In (21), $\hat{\dot{\xi}}^{t,n} = \hat{f}(\xi^{t,n})$ are computed directly from (9).

Both SEDS-Likelihood and SEDS-MSE can be formulated as a Nonlinear Programming (NLP) problem [32] and can be solved using standard constrained optimization techniques. We use a Successive Quadratic Programming (SQP) approach that relies on a quasi-Newton method[7] to solve the constrained optimization problem [32]. SQP minimizes a quadratic approximation of the Lagrangian function over a linear approximation of the constraints.[8]

Our implementation of SQP has several advantages over general purpose solvers. First, we have an analytic expression of the derivatives, improving significantly the performances. Second, our code is tailored to solve the specific problem at hand. For example, a reformulation guarantees that the optimization constraints (20a), (20c), (20d), and (20e) are satisfied. There is, thus, no longer the need to explicitly enforce them during the optimization. The analytical formulation of derivatives and the mathematical reformulation to satisfy the optimization constraints are explained in detail in [33].

Algorithm 1 Procedure to determine an initial guess for the optimization parameters

Input: $\{\xi^{t,n}, \dot{\xi}^{t,n}\}_{t=0,n=1}^{T^n,N}$ and K

1: Run EM over demonstrations to find an estimate of π^k, μ^k, and Σ^k, $k \in 1..K$.
2: Define $\tilde{\pi}^k = \pi^k$ and $\tilde{\mu}_{\xi}^k = \mu_{\xi}^k$
3: Transform covariance matrices such that they satisfy the optimization constraints given by Eq. 20(b) and (c):

$$\begin{cases} \tilde{\Sigma}_{\xi}^k = \boldsymbol{I} \circ abs(\Sigma_{\xi}^k) \\ \tilde{\Sigma}_{\dot{\xi}\xi}^k = -\boldsymbol{I} \circ abs(\Sigma_{\dot{\xi}\xi}^k) \\ \tilde{\Sigma}_{\dot{\xi}}^k = \boldsymbol{I} \circ abs(\Sigma_{\dot{\xi}}^k) \\ \tilde{\Sigma}_{\xi\dot{\xi}}^k = -\boldsymbol{I} \circ abs(\Sigma_{\xi\dot{\xi}}^k) \end{cases} \quad \forall k \in 1..K$$

where $\circ$ and $abs(.)$ corresponds to entrywise product and absolute value function, and $\boldsymbol{I}$ is a $d \times d$ identity matrix.
4: Compute $\tilde{\mu}_{\dot{\xi}}^k$ by solving the optimization constraint given by Eq. 20(a):

$$\tilde{\mu}_{\dot{\xi}}^k = \tilde{\Sigma}_{\dot{\xi}\xi}^k (\tilde{\Sigma}_{\xi}^k)^{-1} (\tilde{\mu}_{\xi}^k - \xi^*)$$

Output: $\boldsymbol{\theta}^0 = \{\tilde{\pi}^1..\tilde{\pi}^K; \tilde{\mu}^1..\tilde{\mu}^K; \tilde{\Sigma}^1..\tilde{\Sigma}^K\}$

Note that a feasible solution to these NLP problems always exists. Algorithm 1 provides a simple and efficient way to compute a feasible initial guess for the optimization parameters. Starting from an initial value, the solver tries to optimize the value of $\boldsymbol{\theta}$ such that the cost function J is minimized. However, since the proposed NLP problem is nonconvex, one cannot ensure to find the globally optimal solution. Solvers are usually very sensitive to initialization of the parameters and will often converge to some local minima of the objective function. Based on our experiments, running the optimization with the initial guess that is obtained from Algorithm 1 usually results in a good local minimum. In all experiments that are reported in Section VI, we ran the initialization three to four times, and use the result from the best run for the performance analysis.

We use the BIC to choose the optimal set K of Gaussians. The BIC determines a tradeoff between optimizing the model's likelihood and the number of parameters that are needed to encode the data

$$\text{BIC} = \mathcal{T} J(\boldsymbol{\theta}) + \frac{n_p}{2} \log(\mathcal{T}) \tag{22}$$

where $J(\boldsymbol{\theta})$ is the normalized log-likelihood of the model that is computed using (19), and n_p is the total number of free parameters. The SEDS-Likelihood approach requires the estimation of $K(1 + 3d + 2d^2)$ parameters (the priors π^k, mean μ^k, and covariance Σ^k are of size 1, $2d$, and $d(2d + 1)$,

[6] In our previous work [5], we used a different MSE cost function, which balanced the effect of following the trajectory and the speed. See Appendix A for a comparison of results using both cost functions and further discussion.

[7] Quasi-Newton methods differ from classical Newton methods in that they compute an estimate of the Hessian function $H(\xi)$ and, thus, do not require a user to provide it explicitly. The estimate of the Hessian function progressively approaches to its real value as optimization proceeds. Among quasi-Newton methods, we use Broyden–Fletcher–Goldfard–Shanno [32].

[8] Given the derivative of the constraints and an estimate of the Hessian and the derivatives of the cost function with respect to the optimization parameters, the SQP method finds a proper descent direction (if it exists) that minimizes the cost function while not violating the constraints. To satisfy equality constraints, SQP finds a descent direction that minimizes the cost function by varying the parameters on the hypersurface that satisfies the equality constraints. For inequality constraints, SQP follows the gradient direction of the cost function whenever the inequality holds (inactive constraints). Only at the hypersurface where the inequality constraint becomes active does SQP look for a descent direction that minimizes the cost function by varying the parameters on the hypersurface or toward the inactive constraint domain.

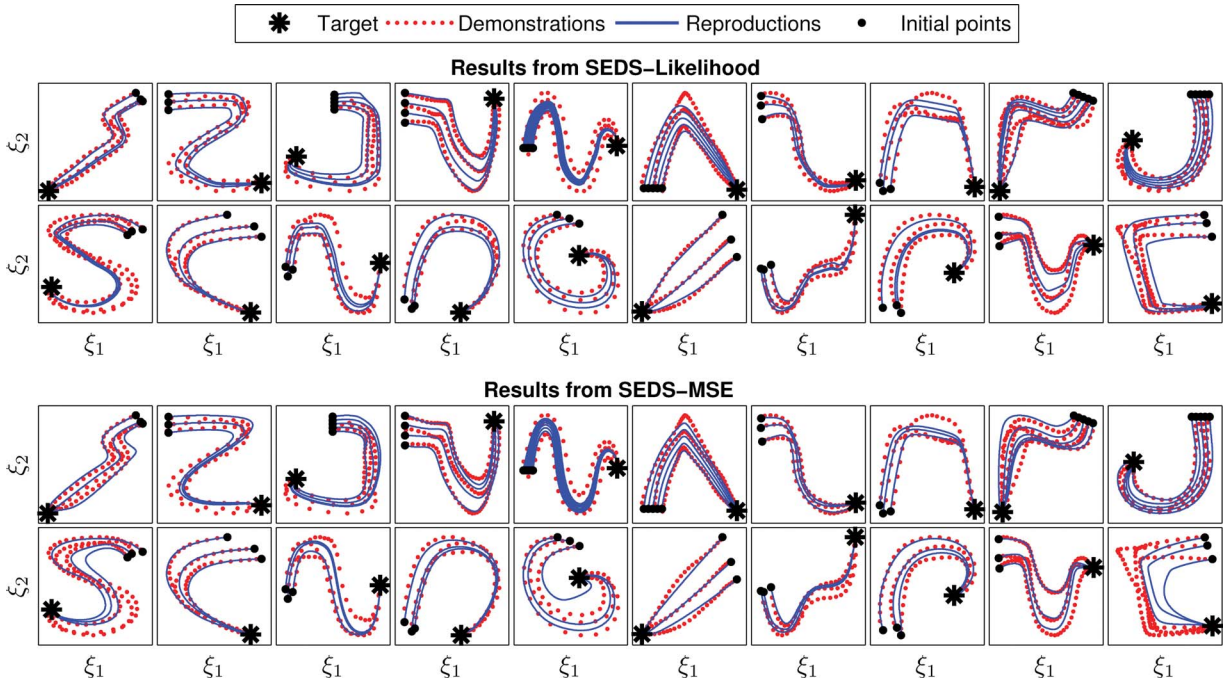

Fig. 6. Performance comparison of SEDS-Likelihood and SEDS-MSE through a library of 20 human handwriting motions.

respectively). However, the number of parameters can be reduced since the constraints given by (20a) provide an explicit formulation to compute μ_{ξ}^{k} from other parameters (i.e., $\mu_{\dot{\xi}}^{k}$, Σ_{ξ}^{k}, and $\Sigma_{\dot{\xi}\xi}^{k}$). Thus, the total number of parameters to construct a GMM with K Gaussians is $K(1+2d(d+1))$. As for SEDS-MSE, the number of parameters is even more reduced since when constructing $\hat{f}$, the term $\Sigma_{\dot{\xi}}^{k}$ is not used and, thus, can be omitted during the optimization. Taking this into account, the total number of learning parameters for the SEDS-MSE reduces to $K(1+\frac{3}{2}d(d+1))$. For both approaches, learning grows linearly with the number of Gaussians and quadratically with the dimension. In comparison, the number of parameters in the proposed method is fewer than GMM and LWPR.[9] The retrieval time of the proposed method is low and in the same order of GMR and LWPR.

The source code of SEDS can be downloaded from http://lasa.epfl.ch/sourcecode/.

VI. Experimental Evaluations

Performance of the proposed method is first evaluated against a library of 20 human handwriting motions. These were chosen as they provide realistic human motions while ensuring that imprecision in both recording and generating motion is minimal. Precisely, in Section VI-A, we compare the performance of the SEDS method when using either the likelihood or MSE. In Section VI-B, we validate SEDS to estimate the dynamics of motion of two robot platforms: 1) the 7-DOF right arm of the humanoid robot iCub and 2) the six DOF industrial robot Katana-T arm. In Sections VI-C and VI-D, we show that the method can learn second- and higher order dynamics that allows us to embed different local dynamics in the same model. Finally, in Section VI-E, we compare our method with those of four alternative methods GMR, LWPR, GPR, and BM.

A. Stable Estimator of Dynamical Systems: Likelihood Versus Mean Square Error

In Section V, we proposed two objective functions: likelihood and MSE for training the SEDS model. We compare the results that are obtained with each method for modeling 20 handwriting motions. The demonstrations are collected from pen input using a Tablet-PC. Fig. 6 shows a qualitative comparison of the estimate of handwriting motions. All reproductions were generated in simulation to exclude the error due to the robot controller from the modeling error. The accuracy of the estimate is measured according to (23), with which the method accuracy in estimating the overall dynamics of the underlying model $\hat{f}$ is quantified by measuring the discrepancy between the direction and magnitude of the estimated and observed velocity vectors for all training data points[10]

[9]The number of learning parameter in GMR and LWPR is $K(1+3d+2d^2)$ and $\frac{7}{2}K(d+d^2)$, respectively.

[10]Equation (23) measures the error in our estimation of both the direction and magnitude of the velocity. It is, hence, a better estimate of how well our model encapsulates the dynamics of the motion, in contrast with an MSE on the velocity magnitude alone.

TABLE I
PERFORMANCE COMPARISON OF SEDS-LIKELIHOOD AND SEDS-MSE IN LEARNING 20 HUMAN HANDWRITING MOTIONS

Method	Average/Range of error $\bar{e}$	Average/Range of No. of Parameters	Average Training Time (sec)
SEDS-MSE	0.25 / [0.18-0.43]	50 / [20 - 70]	27.4 / [3.1-87.7]
SEDS-Likelihood	0.19 / [0.11-0.33]	65 / [26 - 91]	19.2 / [2.2-48.3]

$$\bar{e} = \frac{1}{\mathcal{T}} \sum_{n=1}^{N} \sum_{t=0}^{T^n} \left(r \left(1 - \frac{(\dot{\xi}^{t,n})^T \hat{\dot{\xi}}^{t,n}}{\|\dot{\xi}^{t,n}\| \|\hat{\dot{\xi}}^{t,n}\| + \epsilon} \right)^2 + q \frac{(\dot{\xi}^{t,n} - \hat{\dot{\xi}}^{t,n})^T (\dot{\xi}^{t,n} - \hat{\dot{\xi}}^{t,n})}{\|\dot{\xi}^{t,n}\| \|\hat{\dot{\xi}}^{t,n}\| + \epsilon} \right)^{\frac{1}{2}} \quad (23)$$

where r and q are positive scalars that weigh the relative influence of each factor,[11] and ϵ is a very small positive scalar.

The quantitative comparison between the two methods is represented in Table I. SEDS-Likelihood slightly outperforms SEDS-MSE in accuracy of the estimate, as seen in Fig. 6 and Table I. Optimization with MSE results in a higher value of the error. This could be due to the fact that (21) only considers the norm of $\dot{\xi}$ during the optimization, while when computing $\bar{e}$, the direction of $\dot{\xi}$ is also taken into account [see (23)]. Although one could improve the performance of SEDS-MSE by considering the direction of $\dot{\xi}$ in (21), this would make the optimization problem more difficult to solve by changing a convex objective function into a nonconvex one.

SEDS-MSE is advantageous over SEDS-Likelihood in that it requires fewer parameters (this number is reduced by a factor of $\frac{1}{2}Kd(d+1)$). On the other hand, SEDS-MSE has a more complex cost function that requires computing GMR at each iteration over all training data points. As a result, the use of MSE makes the algorithm computationally more expensive, and it has a slightly longer training time (see Table I).

Following the previous observations that SEDS-Likelihood outperforms SEDS-MSE in terms of accuracy of the reconstruction and the training time, in the rest of the experiments, we will use only SEDS-Likelihood to train the globally stable model.[12]

B. Learning Point-to-Point Motions in the Operational Space

We report on five robot experiments to teach the Katana-T and the iCub robots to perform nonlinear point-to-point motions. In all our experiments, the origin of the reference coordinates system is attached to the target. The motion is, hence, controlled with respect to this frame of reference. Such representation makes the parameters of a DS invariant to changes in the target position.

In the first experiment, we teach a 6-DOF industrial Katana-T arm how to put small blocks into a container[13] (see Fig. 7). We use the Cartesian coordinates system to represent the motions. In order to have human-like motions, the learned model should be able to generate trajectories with both similar position and velocity profiles to the demonstrations. In this experiment, the task was shown to the robot six times and was learned using $K = 6$ Gaussian functions. Fig. 7(a) illustrates the obtained results for generated trajectories starting from different points in the task space. The direction of motion is indicated by arrows. All reproduced trajectories are able to follow the same dynamics (i.e., having similar position and velocity profile) as the demonstrations.

Immediate adaptation: Fig. 7(b) shows the robustness of the model to the change in the environment. In this graph, the original trajectory is plotted in thin blue line. The thick black line represents the generated trajectory for the case where the target is displaced at $t = 1.5$ s. Having defined the motion as autonomous DS, the adaptation to the new target's position can be done instantly.

Increasing accuracy of generalization: While convergence to the target is always ensured from conditions that are given by (13), due to the lack of information for points far from demonstrations, the model may reproduce some trajectories that are not consistent with the usual way of doing the task. For example, consider Fig. 8(a), i.e., when the robot starts the motion from the left side of the target, it first turns around the container and then approaches the target from its right side. This behavior may not be optimal as one expects the robot to follow the shortest path to the target and reach it from the same side as the one it started from. However, such a result is inevitable since the information that is given by the teacher is incomplete, and thus, the inference for points that are far from the demonstrations are not reliable. In order to improve the task execution, it is necessary to provide the robot with more demonstrations (information) over regions that are not covered before. By showing the robot more demonstrations and retraining the model with the new data, the robot is able to successfully accomplish the task [see Fig. 8(b)].

The second and third experiments consisted of having Katana-T robot place a saucer at the center of the tray and putting a cup on the top of the saucer. Both tasks were shown four times and were learned using $K = 4$ Gaussians. The experiments and the generalization of the tasks starting from different points in the space are shown in Figs. 9 and 10. Fig. 11 shows the adaptation of both models in the face of perturbations. Note that in this experiment, the cup task is executed after finishing the saucer task; however, for convenience, we superimpose both tasks in the same graph. In both tasks, the target (i.e., the saucer for the cup task and the tray for the saucer task) is displaced during the execution of the task at the time $t = 2$ s. In both experiments, the adaptation to the perturbation is handled successfully.

[11] Suitable values for r and q must be set to satisfy the user's design criteria that may be task dependent. In this paper, we consider $r = 0.6$ and $q = 0.4$.

[12] Note that in our experiments, the difference between the two algorithms in terms of the number of parameters is small and, thus, is not a decisive factor.

[13] The robot is only taught how to move blocks. The problem of grasping the blocks is out of the scope of this paper. Throughout the experiments, we pose the blocks such that they can be easily grasped by the robot.

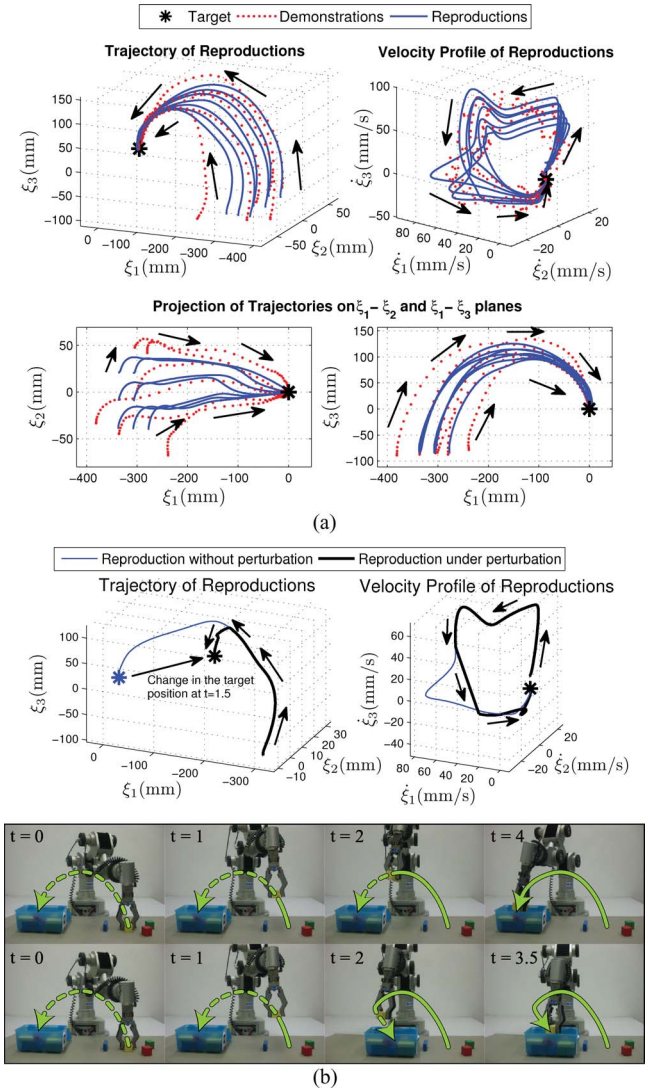

Fig. 7. Katana-T arm that performs the experiment of putting small blocks into a container. See the text for further information. (a) Ability of the model to reproduce similar trajectories starting from different points in the space. (b) Ability of the model to adapt its trajectory on the fly to a change in the target's position.

The fourth and fifth experiments consisted of having the 7-DOF right arm of the humanoid robot iCub perform complex motions, containing several nonlinearities (i.e., successive curvatures) in both position and velocity profiles. Similar to earlier, we use the Cartesian coordinates system to represent these motions. The tasks are shown to the robot by teleoperating it using motion sensors (see Fig. 1). Fig. 12 illustrates the result for the first task where the iCub starts the motion in front of its face. Then, it does a semispiral motion toward its right side, and finally at the bottom of the spiral, it stretches forward its hand completely. In the second task, the iCub starts the motion close to its left forehand. Then, it does a semicircular motion upward and finally brings its arm completely down (see Fig. 13). The two experiments were learned using five and four Gaussian functions, respectively. In both experiments, the robot is able to successfully follow the demonstrations and to gener-

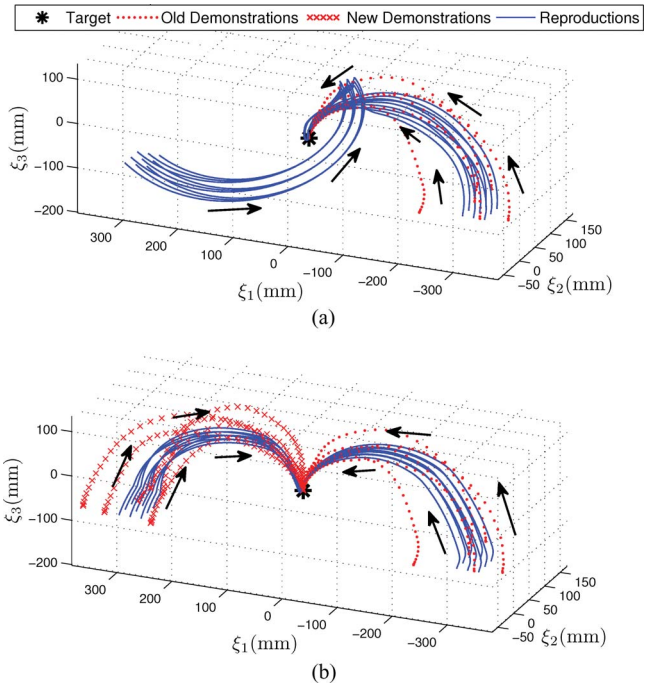

Fig. 8. Improving the task execution by adding more data for regions that are far from the demonstrations. (a) Generalization based on the original model. (b) Generalization after retraining the model with the new data.

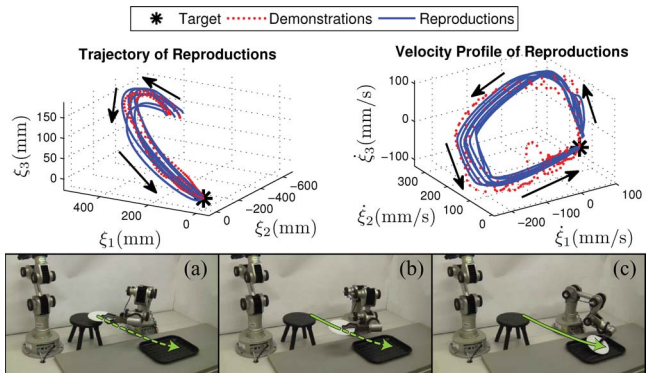

Fig. 9. Katana-T arm that performs the experiment of putting a saucer on a tray.

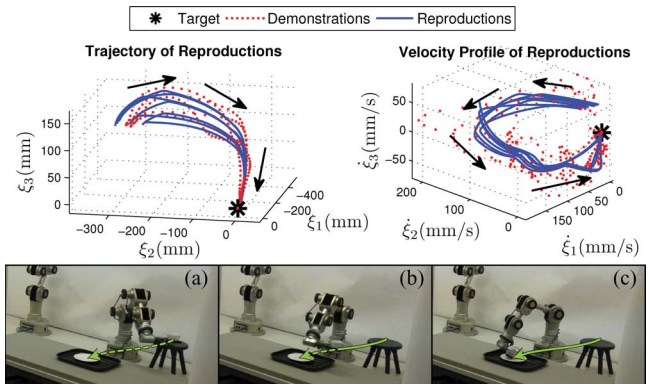

Fig. 10. Katana-T arm that performs the experiment of putting a cup on a saucer.

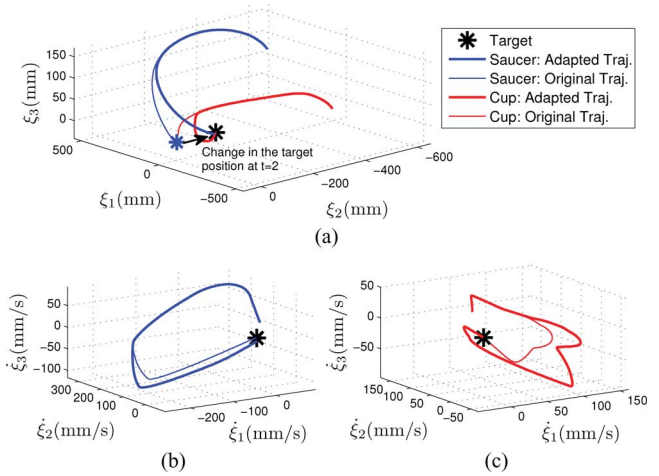

Fig. 11. Ability of the model to on-the-fly adapt its trajectory to a change in the target's position. (a) Trajectory of reproductions. (b) Velocity profile for the saucer task. (c) Velocity profile for the cup task.

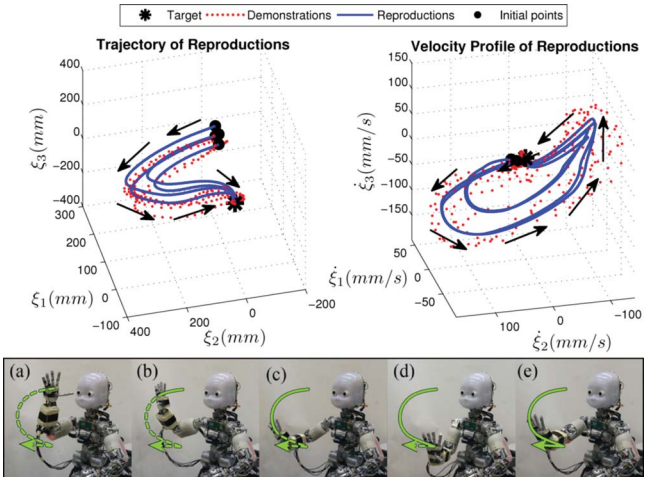

Fig. 12. First experiment with the iCub. The robot does a semispiral motion toward its right side, and at the bottom of the spiral, it stretches forward its hand completely.

alize the motion for several trajectories with different starting points. Similar to what was observed in the three experiments with the Katana-T robot, the models that are obtained for the iCub's experiments are robust to perturbations.

C. Learning Second-Order Dynamics

So far, we have shown how DS can be used to model/learn a demonstrated motion when modeled as a first-order time-invariant ODE. Although this class of ODE functions are generic enough to represent a wide variety of robot motions, they fail to accurately define motions that rely on second-order dynamics such as a self-intersecting trajectory or motions for which the starting and final points coincide with each other (e.g., a triangular motion). Critical to these kinds of motion is the ambiguity in the correct direction of velocity at the intersection point if the model's variable ξ considered to be only the cartesian position (i.e., $\xi = x \Rightarrow \dot{\xi} = \dot{x}$). This ambiguity usually results in skipping the loop part of the motion. However, in this

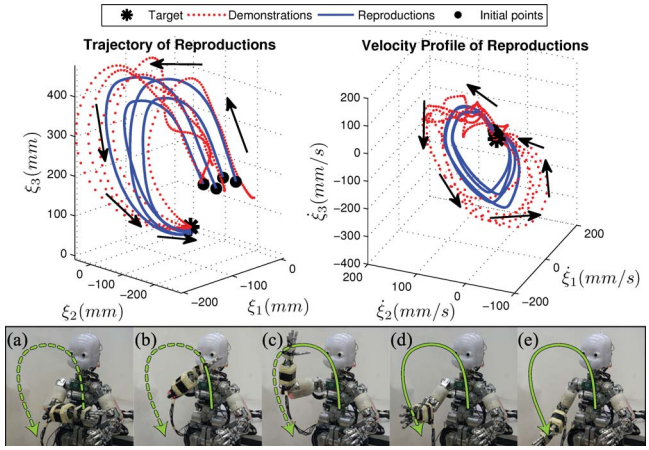

Fig. 13. Second experiment with the iCub. The robot does a semicircle motion upward and brings its arm completely down.

example, this problem can be solved if one defines the motion in terms of position, velocity, and acceleration, i.e., second-order dynamics:

$$\ddot{x} = g(x, \dot{x}) \tag{24}$$

where g is an arbitrary function. Observe that any second-order dynamics in the form of (24) can be easily transformed into a first-order ODE through a change of variable, i.e.,

$$\begin{cases} \dot{x} = v \\ \dot{v} = g(x, v) \end{cases} \Rightarrow [\dot{x}; \dot{v}] = f(x, v) \tag{25}$$

Having defined $\xi = [x; v]$ and, thus, $\dot{\xi} = [\dot{x}; \dot{v}]$, (25) reduces to $\dot{\xi} = f(\xi)$ and, therefore, can be learned with the methods that are presented in this paper. We verify the performance of our method in learning a second-order motion via a robot task. In this experiment, the iCub performs a loop motion with its right hand, where the motion lies in a vertical plane and, thus, contains a self-intersection point (see Fig. 14). Here, the task is shown to the robot five times. The motion is learned with seven Gaussian functions with SEDS-Likelihood. The results demonstrate the ability of SEDS to learn second-order dynamics.

By extension, since any nth-order autonomous ODE can be transformed into a first-order autonomous ODE, the proposed methods can also be used to learn higher order dynamics, however, at the cost of increasing the dimensionality of the system. If the dimensionality of an nth-order DS is d, the dimensionality of the transformed dynamics into a first-order DS is $n \times d$. Hence, increasing the order of the DS is equivalent to increasing the dimension of the data. As the dimension increases, the number of optimization parameters also increases. If one optimizes the values of these parameters that are based on using a quasi-Newton method, the learning problem indeed becomes intractable as the number of dimensions increases. As an alternative solution, one can define the loop motion in terms of both the Cartesian position x and a phase variable. The phase-dependent DS has lower dimension (i.e., dimensionality of $d + 1$) compared with the second-order DS and is more tractable to learn. However, as it is already discussed in Section II, the use of the phase variable

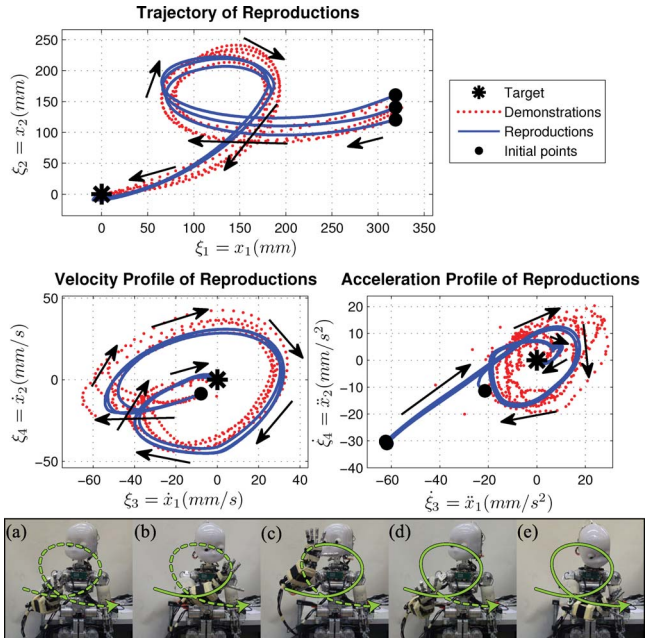

Fig. 14. Learning a self-intersecting motion with a second-order dynamics.

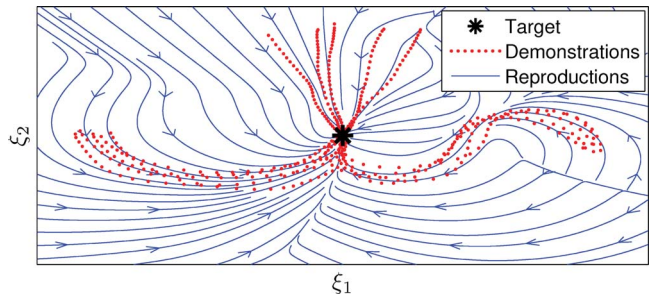

Fig. 15. Embedding different ways of performing a task in one single model. The robot follow an arc, a sine, or a straight line starting from different points in the workspace. All reproductions were generated in simulation.

makes the system time dependent. Depending on the application, one may prefer to choose the system of (25) and learn a more complex DS, or to use its phase variable form, which is time dependent but easier to learn.

D. Encoding Several Motions Into One Single Model

We have so far assumed that a single dynamical system drives a motion; however, sometimes it may be necessary to execute a single task in different manners starting from different areas in the space, mainly to avoid joint limits, task constraints, etc. We have shown an example of such an application in an experiment with the Katana-T robot (see Fig. 8). Now, we show a more complex example and use SEDS-Likelihood to integrate different motions into one single model (see Fig. 15). In this experiment, the task is learned using $K = 7$ Gaussian functions, and the 2-D demonstrations are collected from pen input using a Tablet-PC. The model is learned using SEDS-Likelihood, and it is provided with all demonstration data points at the same time without specifying the dynamics they belong to. Looking at Fig. 15, we see that all the three dynamics are learned successfully with a single model, and the robot is able to approach the target following an arc, a sine function, or a straight line path, respectively, starting from the left, right, or top side of the task space. While reproductions follow locally the desired motion around each set of demonstrations, they smoothly switch from one motion to another in areas between demonstrations.

E. Comparison With Alternative Methods

The proposed method is also compared with three of the best performing regression methods to date (GPR, GMR with EM, and LWPR[14]) and our previous work BM on the same library of handwriting motions that are represented in Section VI-A (see Table II) and the robot experiments that are described in Sections VI-B–D (see Table III). All reproductions were generated in simulation to exclude the error due to the robot controller from the modeling error. Fig. 3 illustrates the difference between these five methods on the estimation of a 2-D motion. To ensure fairer comparison across techniques, GMR was trained with the same number of Gaussians as that found with BIC on SEDS.

As expected, GPR is the most accurate method. GPR performs a very precise nonparametric density estimation and is, thus, bound to give optimal results when using all of the training examples for inference (i.e., we did not use a sparse method). However, this comes at the cost of increasing the computation complexity and storing all demonstration data points (i.e., higher number of parameters). GMR outperforms LWPR by being more accurate and requiring fewer parameters.

Both BM and SEDS-Likelihood are comparatively as accurate as GMR and LWPR. To recall, neither GPR, GMR nor LWPR ensure stability of the system (neither local nor global stability), and BM only ensures local stability (see Section II and Fig. 3). SEDS outperforms BM in that it ensures global asymptotic stability and can better generalize the motion for trajectories far from the demonstrations. In most cases, BM is more accurate (although marginally so). BM offers more flexibility since it unfolds a motion into a set of discrete jointwise partitions and ensures that the motion is locally stable within each partition. SEDS is more constraining since it tries to fit a motion with a single globally stable dynamics. Finally, in contrast with BM, SEDS also enables to encode stable models of several motions into one single model (e.g., see Section VI-D).

VII. Discussion and Future Work

In this paper, we presented a method to learn arbitrary discrete motions by modeling them as nonlinear autonomous DS. We proposed a method that is called *SEDS* to learn the parameters of a GMM by solving an optimization problem under strict stability constraint. We proposed two objective functions that are *SEDS-MSE* and *SEDS-Likelihood* for this optimization problem.

[14]The source code of all the three is downloaded from the website of their authors.

TABLE II
PERFORMANCE COMPARISON OF THE PROPOSED METHODS WITH ALTERNATIVE APPROACHES IN LEARNING 20 HUMAN HANDWRITING MOTIONS

Method	Stability Ensured?	Average / Range of error $\bar{e}$	Average / Range of No. of Parameters	Average / Range of Training Time (sec)	Average Retrieval Time (sec)
SEDS-Likelihood	Yes / Global	0.19 / [0.11 - 0.33]	65 / [26 - 91]	19.23 / [2.22 - 48.26]	0.00068
BM	Yes / Local	0.18 / [0.11 - 0.50]	98 / [56 - 196]	20.4 / [3.56 - 55.16]	0.0019
GMR	No	0.13 / [0.08 - 0.28]	75 / [30 - 105]	0.211 / [0.038 - 0.536]	0.00068
LWPR	No	0.17 / [0.07 - 0.35]	609 / [168 - 1239]	2.49 / [1.51 - 4.01]	0.00054
GPR	No	0.04 / [0.02 - 0.07]	2190 / [1806 - 3006]†	52.42 / [22.84 - 142.27]	0.07809

†GPR's learning parameter is of order of $d(d+1)$; however, it also requires keeping all training data points to estimate the output $\dot{\xi}$. Hence, the total number of required parameters is $d(d+1)+2n\times d$, where n is the total number of data points.

TABLE III
PERFORMANCE COMPARISON OF THE PROPOSED METHODS WITH ALTERNATIVE APPROACHES IN LEARNING ROBOT EXPERIMENTS PRESENTED IN SECTIONS VI-B AND C

Experiment	Error $\bar{e}$					No. of Parameters					Training Time (sec)				
	SEDS	BM	GMR	LWPR	GPR	SEDS	BM	GMR	LWPR	GPR	SEDS	BM	GMR	LWPR	GPR
Katana-Block	1.6355	0.6465	1.4071	1.2619	0.2746	78	81	90	3780	7980	128.5	167.3	1.487	9.082	701
Katana-Saucer	0.2962	0.2285	0.1622	0.2595	0.0167	52	243	60	252	6930	217.1	294.9	0.398	4.017	543.6
Katana-Cup	0.4673	0.3758	0.5886	0.5136	0.1414	52	189	60	1008	7872	91.31	156.7	0.285	6.382	748.4
iCub-Task 1	1.0305	1.4197	1.2950	1.7219	0.0440	65	297	75	840	9642	161.5	317.6	0.833	8.034	834.5
iCub-Task 2	1.6921	1.1434	1.5255	1.8328	0.0487	52	297	60	336	8166	13.6	164.3	0.414	6.354	1786
iCub-2nd Order	0.2381	0.1025	0.0867	0.0625	0.0432	287	1144	315	5250	8020	50.77	191.7	1.986	7.342	436.2
Multi Model	0.26	—	0.1846	0.1626	0.0878	91	—	105	630	9006	58.59	—	2.61	10.82	2026.8

The result for SEDS is obtained using likelihood as the objective function.

The models result from optimizing both objective functions benefit from the inherent characteristics of autonomous DS, i.e., online adaptation to both temporal and spatial perturbation. However, each objective function has its own advantages and disadvantages. Using log-likelihood is advantageous in that it is more accurate and smoother than MSE. Furthermore, the MSE cost function is slightly more time consuming since it requires computing GMR at each iteration for all training data points. However, the MSE objective function requires fewer parameters than the likelihood one which may make the algorithm faster in higher dimensions or when higher number of components is used.

None of the two methods are globally optimal as they deal with a nonconvex objective function. However, in practice, in the 20 handwriting examples and the six robot tasks, which we reported here, we found that SEDS approximation was quite accurate. An assumption made throughout this paper is that represented motions can be modeled with a first-order time-invariant ODE. While the nonlinear function that is given by (9) is able to model a wide variety of motions, the method cannot be used for some special cases that violate this assumption. Most of the time, this limitation can be tackled through a change of variable (as presented in our experiments; see Fig. 14).

The stability conditions at the basis of SEDS are sufficient conditions to ensure global asymptotic stability of nonlinear motions when modeled with a mixture of Gaussian functions. Although our experiments showed that a large library of robot motions can be modeled while satisfying these conditions, these global stability conditions might be too stringent to accurately model some complex motions. For these cases, the user could choose local approaches such as BM to accurately model desired motions.

While, in Section VI-C, we showed how higher order dynamics can be used to model more complicated movements, determining the model order is definitely not a trivial task. It relies on having a good idea of what matters for the task at hand. For instance, higher order derivatives are useful to control for smoothness, jerkiness, and energy consumption and, hence, may be used if the task requires optimizing for such criteria.

Incremental learning is often crucial to allow the user to refine the model in an interactive manner. At this point in time, the SEDS training algorithm does not allow for incremental retraining of the model. If one was to add new demonstrations after training the model, one would have to either retrain entirely the model that is based on the combined set of old and new demonstrations or build a new model from the new demonstrations and merge it with the previous model.[15] For a fixed number of Gaussians, the former usually results in having a more accurate model, while the latter is faster to train (because it only uses the new set of demonstrations in the training).

Ongoing work is directed at designing an online learning version of SEDS whereby the algorithm optimizes the parameters of the model incrementally as the robot explores the space of motion. This algorithm would also allow for the user to provide corrections and, hence, to refine the model locally, along the lines that we followed in [34].

Furthermore, we are currently endowing the method with the on-the-fly ability to avoid possible obstacle(s) during the

[15]Two GMM with K^1 and K^2 number of Gaussian functions can be merged into a single model with $K=K^1+K^2$ Gaussian functions by concatenating their parameters, i.e., $\boldsymbol{\theta}=\{\theta^1\dots\theta^{K^1}\dots\theta^K\}$, where $\theta^k=\{\pi^k,\mu^k,\Sigma^k\}$. The resulting model is no longer (locally) optimal; however, it could be an accurate estimation of both models, especially when there is no overlapping between the two models.

execution of a task. We will also focus on integrating physical constraints of the system (e.g., robot's joints limit, the task's constraint, etc.) into the model to solve for this during our global optimization. Finally, while we have shown that the system could embed more than one motion and, hence, account for different ways to approach the same target, depending on where the motion starts in the workspace, we have still yet to determine how many different dynamics can be embedded in the same system.

VIII. Summary

DS offer a framework that allows for fast learning of robot motions from a small set of demonstrations. They are also advantageous in that they can be easily modulated to produce trajectories with similar dynamics in areas of the workspace that is not covered during training. However, their application to robot control has been given little attention so far, mainly because of the difficulty of ensuring stability. In this paper, we presented an optimization approach for statistically encoding a dynamical motion as a first-order autonomous nonlinear ODE with Gaussian Mixtures. We addressed the stability problem of autonomous nonlinear DS and formulated sufficient conditions to ensure global asymptotic stability of such a system. Then, we proposed two optimization problems to construct a globally stable estimate of a motion from demonstrations.

We compared performance of the proposed method with current widely used regression techniques via a library of 20 handwriting motions. Furthermore, we validated the methods in different point-to-point robot tasks that are performed with two different robots. In all experiments, the proposed method was able to successfully accomplish the experiments in terms of high accuracy during reproduction, the ability to generalize motions to unseen contexts, and the ability to adapt on the fly to spatial and temporal perturbations.

Appendix A

Comparison With the Cost Function Described in [5]

In our previous work [5], we had used a different MSE cost function from that proposed in this paper that balanced the error in both position and velocity

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T^n}\Big(\omega_\xi \|\hat{\xi}^n(t) - \xi^{t,n}\|^2 + \omega_{\dot{\xi}} \|\hat{\dot{\xi}}^n(t) - \dot{\xi}^{t,n}\|^2\Big). \quad (26)$$

$\hat{\dot{\xi}}^n(t) = \hat{f}(\hat{\xi}^n(t))$ are computed directly from (9). $\hat{\xi}^n(t) = \sum_{i=0}^{t} \hat{\dot{\xi}}^n(i)dt$ generate an estimate of the corresponding demonstrated trajectory ξ^n by starting from the same initial points as that demonstrated, i.e., $\hat{\xi}^n(0) = \xi^{0,n}\ \forall n \in 1 \ldots N$. ω_ξ and $\omega_{\dot{\xi}}$ are positive scalars weighing the influence of the position and velocity terms in the cost function.

In contrast with the cost function that is proposed in this paper that assumes independence across data points [see (21)], the aforementioned cost function propagates the effect of the estimation error at each time step along each trajectory. Considering only the error in speed removes these effects (i.e., less complex optimization) while yielding nearly similar performance. For example, when learning the 20 human handwriting motions that are described in Section VI-A, using the cost function given in (26) has yielded an average error of 0.23 (against an error of 0.25 when using only speed in the cost function; see Table I).

TABLE IV
Evaluation of the Effect of Weighting Terms on the MSE Cost Function Presented in [5]

Weights	$\omega_\xi = 1$ $\omega_{\dot{\xi}} = 0$	$\omega_\xi = 1$ $\omega_{\dot{\xi}} = 1$	$\omega_\xi = 0$ $\omega_{\dot{\xi}} = 1$	From normalizing the effect of each term
Motion 1	1.3531	0.4620	**0.3135**	0.5638 $\begin{pmatrix}\omega_\xi = 0.076\\ \omega_{\dot{\xi}} = 0.924\end{pmatrix}$
Motion 2	1.4407	1.0853	1.3161	**0.8652** $\begin{pmatrix}\omega_\xi = 0.293\\ \omega_{\dot{\xi}} = 0.707\end{pmatrix}$
Motion 3	0.3739	**0.3254**	0.8839	0.3853 $\begin{pmatrix}\omega_\xi = 0.101\\ \omega_{\dot{\xi}} = 0.899\end{pmatrix}$
Motion 4	1.6242	1.6047	2.2410	**1.4952** $\begin{pmatrix}\omega_\xi = 0.129\\ \omega_{\dot{\xi}} = 0.871\end{pmatrix}$

Another difficulty that we avoid when considering solely one term in our cost function is related to determining adequate values for the weighting terms (in [5], their value were preset to 1). To illustrate this issue, we ran the optimization on four different handwriting motions by varying the weighting terms given to the velocity and position, respectively. We report on the accuracy as defined in (23) for each of these runs in Table IV. The weighting terms in the fourth column of Table IV are obtained by normalizing the effect of the position and velocity terms. One sees that the advantage of using either of the velocity or position term is not clear cut. For instance, when modeling motion 1, using only the velocity term provides the best result. For motions 2 and 4, the model that is obtained from the normalized weighting terms is more accurate, and the motion 3 is more accurate when the both weights are set equal. In general, it is difficult to say, *a priori*, which weights will result in a more accurate model. This effect is due to the fact that when the weights are changed, the shape of the cost function changes as well in a nonlinear manner.

Acknowledgment

The authors would like to thank J. Peters, for his insightful comments during the development of this paper, and E. Sauser, for preparing the experimental setup with the iCub robot.

References

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cognitive Sci.*, vol. 3, no. 6, pp. 233–242, 1999.
[2] D. Kulic, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains," *Int. J. Robot. Res.*, vol. 27, no. 7, pp. 761–784, 2008.
[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, Cambridge, MA: MIT Press, 2008.
[4] K. Dautenhahn and C. L. Nehaniv, *The Agent-Based Perspective on Imitation*. Cambridge, MA: MIT Press, 2002, pp. 1–40.
[5] S.-M. Khansari-Zadeh and A. Billard, "Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear

programming," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 2676–2683.
[6] E. D. Sontag, "Input to state stability: Basic concepts and results," in *Nonlinear and Optimal Control Theory*, Berlin/Heidelberg, Germany: Springer-Verlag, 2008, ch. 3, pp. 163–220.
[7] J.-H. Hwang, R. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 2, 2003, pp. 1444–1449.
[8] R. Andersson, "Aggressive trajectory generator for a robot ping-pong player," *IEEE Control Syst. Mag.*, vol. 9, no. 2, pp. 15–21, Feb. 1989.
[9] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robot. Auton. Syst.*, vol. 54, pp. 409–413, 2006.
[10] A. Ude, "Trajectory generation from noisy positions of object features for teaching robot paths," *Robot. Auton. Syst.*, vol. 11, no. 2, pp. 113–127, 1993.
[11] M. Muehlig, M. Gienger, S. Hellbach, J. Steil, and C. Goerick, "Task level imitation learning using variance-based movement optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 1177–1184.
[12] K. Yamane, J. J. Kuffner, and J. K. Hodgins, "Synthesizing animations of human manipulation tasks," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 532–539, 2007.
[13] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *Proc. 25th Int. Conf. Mach. Learning*, 2008, pp. 144–151.
[14] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1463–1467, Dec. 2008.
[15] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. New York: Springer-Verlag, 2006.
[16] S. Schaal, C. Atkeson, and S. Vijayakumar, "Scalable locally weighted statistical techniques for real time robot learning," *Appl. Intell.—Spec. Issue Scalable Robot. Appl. Neural Netw.*, vol. 17, no. 1, pp. 49–60, 2002.
[17] A. Dempster and N. L. D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.
[18] S.-M. Khansari-Zadeh and A. Billard, "BM: An iterative algorithm to learn stable non-linear dynamical systems with Gaussian Mixture Models," in *Proc. Int. Conf. Robot. Autom.*, 2010, pp. 2381–2388.
[19] J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[20] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. Int. Conf. Robot. Autom.*, 2009, pp. 1293–1298.
[21] E. Gribovskaya, S. M. Khansari-Zadeh, and A. Billard, "Learning nonlinear multivariate dynamics of motion in robotic manipulators," *Int. J. Robot. Res.*, vol. 30, pp. 1–37, 2010.
[22] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard, "Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 44–54, Jun. 2010.
[23] E. Gribovskaya and A. Billard, "Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots*, 2009, pp. 472–477.
[24] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.
[25] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, pp. 461–464, 1978.
[26] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automat. Control*, vol. AC-19, no. 6, pp. 716–723, Dec. 1974.
[27] D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. van der Linde, "Bayesian measures of model complexity and fit," *J. R. Statist. Soc., Series B (Statist. Methodol.)*, vol. 64, pp. 583–639, 2002.
[28] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Local Gaussian process regression for real time online model learning and control," in *Proc. Adv. Neural Inf. Process. Syst. 21*, 2009, pp. 1193–1200.
[29] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. Adv. Neural Inf. Process. Syst. 18*, 2006, pp. 1257–1264.
[30] M. Seeger, C. K. I. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian Process Regression," in *Proc. 9th Int. Workshop Artif. Intell. Statist.*, 2003.
[31] D. Cohn and Z. Ghahramani, "Active learning with statistical models," *Artif. Intell. Res.*, vol. 4, pp. 129–145, 1996.
[32] M. S. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. New York: Wiley, 2006.
[33] S.-M. Khansari-Zadeh and A. Billard. (2010). "The derivatives of the SEDS optimization cost function and constraints with respect to its optimization parameters," Ecole Polytech. Fed. Lausanne, Lausanne, Switzerland, Tech. Rep., [Online]. Available: http://lasa.epfl.ch/publications/publications.php
[34] B. D. Argall, E. Sauser, and A. Billard, "Tactile guidance for policy refinement and reuse," in *Proc. 9th IEEE Int. Conf. Development Learning*, 2010, pp. 7–12.

S. Mohammad Khansari-Zadeh received the B.Sc. degree in aerospace engineering and the M.Sc. degree in flight dynamics and control from the Department of Aerospace Engineering, Sharif University of Technology, Tehran, Iran, in 2005 and 2008, respectively. He is currently working toward the Ph.D. degree in robotics with the School of Engineering, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland.

His research interests include modeling, control, and stability analysis of nonlinear dynamics and using imitation learning to infer control policies for robot control.

Aude Billard received the B.Sc. and M.Sc. degrees in physics from the Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland, in 1994 and 1995, respectively, and the Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K., in 1998.

She is currently an Associate Professor and the Head of the Learning Algorithms and Systems Laboratory, School of Engineering, EPFL.