Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models

S. Mohammad Khansari-Zadeh, Member, IEEE, Aude Billard, Member, IEEE

Abstract—This paper presents a method for learning arbitrary discrete 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, called Stable Estimator of Dynamical Systems (SEDS), to learn the parameters of the DS to ensure that all motions follow closely the demonstrations while ultimately reaching in 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 encountered during the motion. The method is evaluated through a set of robotic experiments and on a library of human handwriting motions.

Index Terms—Dynamical system, stability, point-to-point motion, statistical learning, programming by demonstration.

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 to 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 guiding 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. Following [5], 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.

A. Formalism

We formulate the encoding of point-to-point motion as control law driven by an autonomous dynamical system: Consider

\[ \dot{\xi} = f(\xi) + \epsilon \]

(1)

where \( f : \mathbb{R}^d \rightarrow \mathbb{R}^d \) is a nonlinear continuous and continuously differentiable function with a single equilibrium point \( \xi^* = f(\xi^*) = 0 \), \( \theta \) is the set of parameters of \( f \), and \( \epsilon \) represents a zero mean additive Gaussian noise. The noise term \( \epsilon \) encapsulates both inaccuracies in sensor measurements and errors resulting from imperfect demonstrations. The function \( f(\xi) \) can be described by a set of parameters \( \theta \), in which the optimal values of \( \theta \) 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 the paper. Our noise-free estimate will thus be:

\[ \dot{\xi} = \hat{f}(\xi) \]

(2)

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

Use of DS is advantageous in that it enables a robot to adapt its trajectory instantly in the face of perturbations [27]. A controller driven by a DS is robust to perturbations because it embeds all possible solutions to reach a target into one single function \( f \). Such a function represents a global map which 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 which result from a sudden displacement in

\(^1\)Assuming a zero mean distribution for the noise makes it possible to estimate the noise free model through regression.
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 kinematic coordinate systems (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 generating 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) \(^7\) [7], 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, not necessarily accurate\(^3\), that makes the inner loop ISS. Hence we focus our efforts on the 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 \(\theta\) of the DS given in Eq. 2. Learning proceeds as an optimization under constraint 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 \(\xi^*\) if \(f(\xi^*) = 0\) and \(\forall \xi^0 \in \mathbb{R}^d\), the generated motion converges asymptotically to \(\xi^*\), i.e.

\[
\lim_{t \to \infty} \xi^t = \xi^* \quad \forall \xi^0 \in \mathbb{R}^d
\]  

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

Non-linear dynamical systems 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 of estimating \(f\) and its parameters \(\theta\) as an optimization under constraint 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 Gaussians. In Section IV we develop conditions for ensuring 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 for estimating the dynamics of motions a) against a library of human writing motions, and b) in two different robotic platforms (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.

## 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 \([8]–[12]\). 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 for segmenting and aligning 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 based on Gaussian kernels \([2], [13], [14]\). These methods provide powerful means for encoding arbitrary multi-dimensional nonlinear trajectories. However, similar to spline-encoding, these ap-
approaches depend on explicit time-indexing and virtually operate in an open-loop. Time dependency makes these techniques very sensitive to both temporal and spatial perturbations. To compensate for this deficiency\(^4\), one requires a heuristic to re-index the new trajectory in time, while simultaneously optimizing a measure of how good the new trajectory follows the desired one. Finding a good heuristic is highly task-dependent and a non-trivial task, and becomes particularly non-intuitive in high-dimensional state spaces [15].

Reference [16] proposed an EM algorithm which 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 for learning complex motions, it is time dependent and thus shares the disadvantages mentioned above.

Dynamical systems (DS) have been advocated as a powerful alternative to modeling robot motions [6], [15]. Existing approaches to the statistical estimation of \( f \) in Eq. 2 use either Gaussian Process Regression (GPR) [17], Locally Weighted Projection Regression (LWPR) [18], or Gaussian Mixture Regression (GMR) [13], [19] where the parameters of the Gaussian Mixture are optimized through Expectation Maximization (EM) [20]. GMR and GPR find a locally optimal model of \( f \) by maximizing the likelihood that the complete model represents the data well, while LWPR minimizes the mean-square error between the estimates and the data (for detailed discussion on these methods see [21]).

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 \( f \) (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 [22]. Fig. 3 illustrates an example of unstable estimation of a nonlinear DS using the above three methods for learning a two-dimensional motion. Fig. 3(a) represents the stability analysis of the dynamics learnt 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.

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 not a 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 locate the region of attraction of the desired target which may never exist, or be very narrow.

The Dynamic Movement Primitives (DMP) offered an interesting and novel approach to the estimation of nonlinear dynamical systems [6]. It was also extended for obstacle avoidance [23]. DMP however relies on a secondary linear dynamical system to ensure the DS’s stability. The modulation between \( f \) and the stable linear dynamics is controlled with a so-called phase variable that creates an implicit time-dependency. The implicit time dependency of DMP makes the system sensitive to both temporal and spatial perturbations, as we show here. Note that this implicit time-dependency remains even in the recent reformulation of DMP suggested by [23]. The time dependency is conveyed through the canonical variable, that acts as a clock for the system. To adapt to changes in the motions duration associated with different initial positions or significant spatial perturbations, one must use a heuristic to re-set the canonical variable. Failing this, the

\(^4\)If one is to model solely 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. Note that the majority of everyday motions are not to be performed in a fixed amount of time, and hence a time-independent encoding may offer a more generic solution to a larger class of motions.
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. Therefore, this dependency on the phase variable may result in undesirable behaviors which were highlighted above.

Fig. 4 illustrate the side-effects of DMP’s time-dependency when modeling a 2-D motion. Motion consists in a sine curve. 500 noise-free training data points were generated from sampling uniformly the function, i.e. \( \xi_1 = [-2\pi, 0] \) and \( \xi_2 = \sin \xi_1 \). We used the most recent version of DMP presented in [23] to learn the motion. Fig. 4(a) and (b) show the ability of DMP to generate similar motions to the demonstration for five starting points close to the demonstration’s starting point. In Fig. 4(c), we perturbed the motion generated by DMP by virtually stopping the system at the perturbation point for 300 millisecond. The results verify the sensitivity of the model to temporal perturbations. While the motion does reach the target, it does not follow the demonstrated profile of motion. Fig. 4(d) shows the generated trajectories starting from one of the demonstrated datapoints. Here, though it is expected that the motion would follow the demonstration toward the target, due to its time-dependency it first departs from demonstration and then approaches it after doing a loop. Fig. 4(e) represents the same result for a starting point very close to the target.

In our prior work [19], we developed a hybrid controller composed of two DSs working concurrently in end-effector and joint angle spaces, resulting in a controller that has no singularities. While this approach was able to adapt on-line 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.

In our previous work [24], we considered an alternative DS approach based on Hidden Markov Model (HMM) and GMR. The method presented there is time-independent and thus robust to temporal perturbations. Asymptotic stability could however not be ensured. Sole a brief verification to avoid large unstabilities was done by evaluating the eigenvalues of each linear DS and ensuring they all have negative real parts. As stated in [24] 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. 6.

In [25], [26] we proposed a heuristics to build iteratively a locally stable estimate of nonlinear DSs. This heuristics requires one to increase the number of Gaussians and retrain the mixture using Expectation-Maximization 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 did not give any 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 [21], we proposed an iterative method, called Binary Merging (BM), to construct a mixture of Gaussians so as to ensure local asymptotic stability at the target, hence the model can solely applied in a region close to demonstrations (see Fig. 3(d)). Though this works provided sufficient conditions to make DS locally stable, similarly to [26], 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 DSs. This approach provides a sound ground for the estimation of nonlinear dynamical systems which is not heuristic driven and has thus the potential for much larger sets of applications, such as the estimation of second order dynamics and for control of multi-degrees of freedom robots as we demonstrate here. Fig. 3(e) represents results 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 without any need to re-index or re-scale. Note that the stability analysis presented here was published in a preliminary form in [27]. The present paper extends largely this work by a) having a more depth discussion on stability, b) by proposing two objective functions to learn parameters of DS and comparing...
their pros and cons c) by having a more detailed comparison of the performance of the proposed method, BM, and three best regression methods for estimating motion dynamics, namely GMR, LWPR, and GPR, and d) by having more robotic 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 [28], 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) [29], the Akaike information criterion (AIC) [30], the deviance information criterion (DIC) [31], that penalize large increase in the number of parameters when this offers only a small gain in the likelihood of the model.

While non parametric methods, such as Gaussian Process or variants on these, offer optimal regression [17], [32], they suffer from the curse of dimensionality. Indeed, computing the estimate regressor $f$ 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 perturbation. There exists various sparse techniques to reduce the sensitivity of these methods to the number of datapoints. However, these techniques either become parametric by predetermining the optimal number of datapoints [33], or they rely on a heuristic such as information gain for determining the optimal subset of datapoints [34]. 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 $f$ become the prior $\pi^k$, the mean $\mu^k$ and the covariance matrices $\Sigma^k$ of the $k = 1..K$ Gaussian functions (i.e. $\theta^k = \{\pi^k, \mu^k, \Sigma^k\}$ and $\theta = \{\theta^1..\theta^K\}$). The mean and the covariance matrix of a Gaussian $k$ are defined by:

$$
\mu^k = \begin{pmatrix} \mu^k_1 \\ \mu^k_2 \\ \vdots \\ \mu^k_D \end{pmatrix} \quad \text{and} \quad \Sigma^k = \begin{pmatrix} \Sigma^k_{11} & \Sigma^k_{12} & \cdots & \Sigma^k_{1D} \\ \Sigma^k_{21} & \Sigma^k_{22} & \cdots & \Sigma^k_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma^k_{D1} & \Sigma^k_{D2} & \cdots & \Sigma^k_{DD} \end{pmatrix}
$$

(4)

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

$$
P(\xi^{t_n, n}, \xi^{t_n, n}; \theta) = \sum_{k=1}^{K} P(k)P(\xi^{t_n, n}|k) \quad \forall n \in 1..N \quad \forall t \in 0..T^n
$$

(5)

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

$$
P(\xi^{t_n, n}, \xi^{t_n, n}|k) = N(\xi^{t_n, n}, \xi^{t_n, n}|\mu^k, \Sigma^k) = \frac{1}{\sqrt{(2\pi)^D|\Sigma^k|}} e^{-\frac{1}{2}[(\xi^{t_n, n} - \mu^k)^T(\Sigma^k)^{-1}(\xi^{t_n, n} - \mu^k)]}
$$

(6)

Substituting Eq. 8 into Eq. 7 yields:

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

(9)

Taking the posterior mean estimate of $P(\dot{\xi}|\xi)$ yields (as described in [35]):

$$
\dot{\xi} = \sum_{k=1}^{K} \frac{P(k)P(\xi|k)}{\sum_{i=1}^{K} P(i)P(\xi|i)} (\mu^k + \Sigma^k (\Sigma^k)^{-1}(\xi - \mu^k))
$$

(7)

The notation of Eq. 7 can be simplified through a change of variable. Let us define:

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

(8)

Substituting Eq. 8 into Eq. 7 yields:

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

(9)

First observe that $\hat{f}$ is now expressed as a nonlinear sum of linear dynamical systems. Fig. 5 illustrates the parameters defined in Eq. 8 and their effects on $\hat{f}(\xi)$ for a 1-D model constructed with 3 Gaussians. Please refer to the text for further information.

![Fig. 5. Illustration of parameters defined in Eq. 8 and their effects on $\hat{f}(\xi)$ for a 1-D model constructed with 3 Gaussians. Please refer to the text for further information.](image.png)

IV. STABILITY ANALYSIS

Stability analysis of dynamical systems 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 [22], where a linear DS can be written as:

$$
\dot{\xi} = A\xi + b
$$

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

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

\[
\begin{align*}
\Sigma_1^\xi &= \Sigma_2^\xi = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}, \\
\Sigma_1^{\xi \xi} &= \begin{bmatrix} -3 & -30 \\ -30 & -3 \end{bmatrix}, \\
\mu_1^\xi &= \mu_2^\xi = \mu_1^\xi = \mu_2^\xi = 0
\end{align*}
\]

Using Eq. 8 we have:

\[
\begin{align*}
A_1 &= \begin{bmatrix} -1 & -10 \\ 1 & -1 \end{bmatrix}, \\
A_2 &= \begin{bmatrix} -1 & 1 \\ -10 & -1 \end{bmatrix}
\end{align*}
\]

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 system. However, the nonlinear combination of the two matrices as per Eq. 9 is stable solely when $\xi_2 = \xi_1$, and is unstable in $\mathbb{R}^d \setminus \{ (\xi_2, \xi_1) | \xi_2 = \xi_1 \}$ (see Fig. 6).

Next we determine sufficient conditions to ensure global asymptotic stability of a series of nonlinear dynamical systems given by Eq. 7.

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

\[
\begin{align*}
(a) & \quad b^k = -A^k\xi^* \\
(b) & \quad A^k + (A^k)^T < 0 \quad \forall k = 1..K
\end{align*}
\]

where $(A^k)^T$ is the transpose of $A^k$, and $< 0$ 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 [22]:

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

\[
\begin{align*}
(a) & \quad V(\xi) > 0 \quad \forall \xi \in \mathbb{R}^d \quad \& \quad \xi \neq \xi^* \\
(b) & \quad \dot{V}(\xi) < 0 \quad \forall \xi \in \mathbb{R}^d \quad \& \quad \xi \neq \xi^* \\
(c) & \quad V(\xi^*) = 0 \quad \& \quad \dot{V}(\xi^*) = 0
\end{align*}
\]

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

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

\[
V(\xi) = \frac{1}{2}(\xi - \xi^*)^T(\xi - \xi^*) \quad \forall \xi \in \mathbb{R}^d
\]

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

\[
\dot{V}(\xi) = \frac{dV}{dt} = \frac{dV}{d\xi} \frac{d\xi}{dt} = \frac{1}{2} \frac{d}{d\xi} \left( (\xi - \xi^*)^T(\xi - \xi^*) \right) \dot{\xi} = (\xi - \xi^*)^T \dot{\xi} = (\xi - \xi^*)^T f(\xi)
\]

\[
= (\xi - \xi^*)^T \sum_{k=1}^{K} h_k(\xi)(A_k^k \xi + b_k)
\]

\[
= (\xi - \xi^*)^T \sum_{k=1}^{K} h_k(\xi)(A_k^k \xi + b_k) = 0 \quad \text{(see Eq. 13-a)}
\]

\[
= (\xi - \xi^*)^T \sum_{k=1}^{K} h_k(\xi)A_k^k(\xi - \xi^*)
\]

\[
= \sum_{k=1}^{K} h_k(\xi)(\xi - \xi^*)^T A_k^k(\xi - \xi^*)
\]

\[
< 0 \quad \forall \xi \in \mathbb{R}^d \quad \& \quad \xi \neq \xi^*
\]

Conditions given by Eq. 14.c is satisfied when substituting $\xi = \xi^*$ into Eqs. 15 and 16:

$^5$A $d \times d$ real symmetric matrix $A$ is positive definite if $\xi^T A \xi > 0$ for all non-zero vectors $\xi \in \mathbb{R}^d$, where $\xi^T$ denotes the transpose of $\xi$. Conversely $A$ is negative definite if $\xi^T A \xi < 0$. For a non-symmetric matrix, $A$ is positive (negative) definite if and only if its symmetric part $\bar{A} = (A + A^T)/2$ is positive (negative) definite.
\[ V(\xi^*) = \frac{1}{2}(\xi - \xi^*)^T(\xi - \xi^*) \bigg|_{\xi = \xi^*} = 0 \] (17)

\[ \dot{V}(\xi^*) = \sum_{k=1}^{K} h^k(\xi - \xi^*)^T A^k(\xi - \xi^*) \bigg|_{\xi = \xi^*} = 0 \] (18)

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

Conditions (a) and (b) in Eq. 13 are sufficient to ensure that an arbitrary nonlinear function given by Eq. 9 is globally asymptotically stable at the target \( \xi^* \). 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 ASYMPOTICALLY STABLE MODELS

Section IV provided us with sufficient conditions whereby the estimate \( \dot{f}(\xi) \) is globally asymptotically stable at the target. It remains now to determine a procedure for computing unknown parameters of Eq. 9, i.e. \( \theta = \{\pi^1, \ldots, \pi^K; \mu^1, \ldots, \mu^K; \Sigma^1, \ldots, \Sigma^K\} \) such that the resulting model is globally asymptotically stable. In this section we propose a learning algorithm, called Stable Estimator of Dynamical Systems (SEDS), that computes optimal values of \( \theta \) 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) Mean Square Error (MSE). The results from both approaches will be evaluated and compared in Section VI-A.

SEDS-LIKELIHOOD: using log-likelihood as a means to construct a model,

\[ \min_{\theta} J(\theta) = -\sum_{n=1}^{N} \sum_{t=0}^{T^n} \log \mathcal{P}(\xi^{t,n}, \xi^{t,n} | \theta) \] (19)

subject to

\[ \begin{align*}
(a) \quad b^k &= -A^k\xi^* \\
(b) \quad A^k + (A^k)^T &< 0 \\
(c) \quad \Sigma^k &> 0 \quad \forall k \in 1..K \\
(d) \quad 0 < \pi^k &\leq 1 \\
(e) \quad \sum_{k=1}^{K} \pi^k & = 1
\end{align*} \] (20)

where \( \mathcal{P}(\xi^{t,n}, \xi^{t,n} | \theta) \) is given by Eq. 5. The first two constraints in Eq. 20 are stability conditions from Section IV. The last three constraints are imposed by the nature of the Gaussian Mixture Model to ensure that \( \Sigma^k \) are positive definite matrices, priors \( \pi^k \) are positive scalars smaller or equal than one, and sum of all priors is equal to one (because the probability value of Eq. 5 should not exceed 1).

SEDS-MSE: using Mean Square Error as a means to quantify the accuracy of estimations based on demonstrations.\(^6\)

\[ \min_{\theta} J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T^n} \|\hat{\xi}^{t,n} - \xi^{t,n}\|^2 \] (21)

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

Both SEDS-Likelihood and SEDS-MSE correspond to a Non-linear Programming (NLP) problem [36] that can be solved using different optimization techniques such as Newton and quasi-Newton algorithms [36], Dynamic Programming [37], etc. In this paper we use a Successive Quadratic Programming (SQP) technique that employs a quasi-Newton method to solve the constrained optimization problem. 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-Goldfarb-Shanno (BFGS) which is one of the most popular quasi-Newton method currently known [36]. Thus, given an estimate of the Hessian function, the derivatives of the cost function and the constraints 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 improve the optimization performance, we also use a line search method [38] to adaptively change the magnitude of each step to obtain an acceptable decrease in the objective function:

\[ \theta^{i+1} = \theta^i + \alpha(\nabla \theta)^i \] (22)

where \( \alpha \) is the line-search’s parameter, \( (\nabla \theta)^i \) corresponds to the appropriate descent direction that minimizes the objective function under the given constrains, and \( i \) is the iteration step.

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 \( \theta \) such that the cost function \( J(\theta) \) is minimized. However since the proposed NLP problem is non-convex, one cannot ensure to find the globally optimal solution. Solvers are usually very sensitive to initialization of the parameters and

\(^6\)In our previous work [27], we suggested a different MSE cost function:

\[ \min_{\theta} J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T^n} \left( \|\xi^n(t) - \xi^{t,n}\|^2 + \|\hat{\xi}^{t,n}(t) - \xi^{t,n}\|^2 \right) \]

where \( \hat{\xi}^{t,n}(t) = \hat{f}(\xi^{t,n}(t)) \) are computed directly from Eq. 9, and \( \xi^n(t) = \sum_{i=0}^{t} \xi^n(i)dt \) generate an estimate of the corresponding demonstrated trajectory \( \xi^n \) by starting from the same initial points as that demonstrated, i.e. \( \xi^n(0) = \xi^{0,n}, \forall n \in 1..N \). In contrast to the cost function proposed in this paper that assumes the independency across datapoints, the above cost function propagates the effect of estimation error at each time step along each trajectory. The independency assumption makes the cost function less complex (faster convergence), while showing similar performance (see Table I in [27] and this paper).
Algorithm 1 Procedure to obtain an initial guess for the optimization parameters

**Input:** \{\( \xi_{t,n}^{k}, \xi_{t,n}^{k} \)\}_{n=1}^{N} and \( K \)

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

\[
\begin{align*}
\tilde{\Sigma}_{k} & = I \circ \text{abs}(\Sigma_{k}) \\
\tilde{\Sigma}_{k}^{*} & = -I \circ \text{abs}(\Sigma_{k}^{*}) \\
\Sigma_{k} & = I \circ \text{abs}(\Sigma_{k}) \\
\Sigma_{k}^{*} & = -I \circ \text{abs}(\Sigma_{k}^{*})
\end{align*}
\]

∀ \( k \in 1..K \)

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

\[
\tilde{\mu}_{k}^{*} = \tilde{\Sigma}_{k}^{\dagger}(\tilde{\Sigma}_{k}^{*})^{-1}(\tilde{\mu}_{k} - \xi^{*})
\]

**Output:** \( \theta^{0} = \{ \pi^{1..K}, \tilde{\mu}^{1..K}, \tilde{\Sigma}^{1..K} \} \)

will often converge to some local minimum of the objective function. Based on our experiments, running the optimization with the initial guess obtained from Algorithm 1 usually results in a good local minimum. In all experiments 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 Bayesian Information Criterion (BIC) to choose the optimal set \( K \) of Gaussians. BIC determines a tradeoff between optimizing the model’s likelihood and the number of parameters needed to encode the data:

\[
BIC = J(\theta) + \frac{n_{p}}{2} \log({\sum_{n=1}^{N} T^{n}}) \tag{23}
\]

where \( J(\theta) \) is the log-likelihood of the model computed using Eq. 19, and \( n_{p} \) is the total number of free parameters. The SEDS-Likelihood approach requires the estimation of \( K(2d^{2} + 3d + 1) \) parameters (the priors \( \pi^{k}, \) mean \( \mu^{k} \) and covariance \( \Sigma^{k} \) are of size \( 1, 2d \) and \( d(2d + 1) \) respectively). However, the number of parameters can be reduced since the constraints given by Eq. 20-(a) provide an explicit formulation to compute \( \mu_{k}^{*} \) from other parameters (i.e. \( \mu_{k}^{*}, \Sigma_{k}^{*}, \) and \( \Sigma_{k}^{*} \)). 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 \( f \), the term \( \Sigma_{k}^{*} \) 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 \( K(1 + \frac{3}{2}(d + 1)) \). 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\(^7\). 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/](http://lasa.epfl.ch/sourcecode/)

VI. EXPERIMENTAL EVALUATIONS

Performance of the proposed methods against competitive methods are first evaluated against a library of 20 handwriting human 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 performance of the SEDS method when using either the likelihood or MSE. In Sections VI-B, we validate SEDS to estimate the dynamics of motion of two robotics platforms (a) the seven degrees of freedom (DOF) right arm of the humanoid robot iCub, and (b) the six DOF industrial robot Katana-T arm. In Sections VI-C and VI-D, we show that the method can learn 2nd and higher order dynamics and that allows to embed different local dynamics in the same model. Finally In Section VI-E, we compare our method against those of four alternative methods GMR, LWPR, GPR, and BM.

A. SEDS training: Likelihood vs. MSE

In Section V we proposed two objective functions: likelihood and MSE for training the SEDS model. We compare the result obtained with each method for modeling 20 handwriting motions. Fig. 7 shows a qualitative comparison of the estimate of handwriting motions. The accuracy of the estimate is measured according to Eq. 24, 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\(^8\).

\[
\bar{e} = \frac{1}{N} \sum_{n=1}^{N} \left( \frac{1}{T^{n}} \sum_{t=0}^{T^{n}} r^{1 - \frac{(\xi_{t,n}^{*} - \hat{\xi}_{t,n}^{*})^{2}}{||\xi_{t,n}^{*}|| ||\xi_{t,n}^{*}|| + \epsilon}} + q^{(\xi_{t,n}^{*} - \hat{\xi}_{t,n}^{*})^{2} ||\xi_{t,n}^{*}|| ||\xi_{t,n}^{*}|| + \epsilon} \right)^{\frac{1}{2}} \tag{24}
\]

where \( r \) and \( q \) are positive scalars that weigh the relative influence of each factor\(^9\), and \( \epsilon \) 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. 7 and Table I. Optimization with MSE results in a higher value of the error. This could be due to the fact that Eq. 21 only considers the norm of \( \xi \) during the optimization, while when computing

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

\(^8\)Eq. 24 measures the error in our estimation of both the direction and magnitude of the motion. It is hence a better estimate of how well our model encapsulates the dynamics of the motion, in contrast to a MSE on the trajectory alone.

\(^9\)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 \( q = 0.4 \).


\[ \xi_1 \]

Results from SEDS−Likelihood

\[ \xi_1 \]

Results from SEDS−MSE

\[ \xi_1 \]

Fig. 7. Performance comparison of SEDS-Likelihood and SEDS-MSE through a library of 20 different human handwriting motions.

Table I

<table>
<thead>
<tr>
<th>Method</th>
<th>Average/Range of error ( \bar{e} )</th>
<th>Average/Range of No. of Parameters</th>
<th>Average Training Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEDS-MSE</td>
<td>0.27 / [0.15-0.49]</td>
<td>50 / [20 - 70]</td>
<td>13.2 / [1.02-20.8]</td>
</tr>
<tr>
<td>SEDS-Likelihood</td>
<td>0.19 / [0.11-0.33]</td>
<td>65 / [26 - 91]</td>
<td>10.6 / [1.16-20.1]</td>
</tr>
</tbody>
</table>

Although one could improve the performance of SEDS-MSE by considering the direction of \( \xi \) in Eq. 21, this would make the optimization problem more difficult to solve by changing a convex objective function into a non-convex one.

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

Following the above 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 used solely SEDS-Likelihood to train the global model.\(^{10}\)

\( \bar{e} \) the direction of \( \xi \) is also taken into account (see Eq. 24).

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

We report on five robotic experiments to teach the Katana-T and the iCub robot 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 six degrees of freedom (DOF) industrial Katana-T arm how to put small blocks into a container\(^{11}\) (see Fig. 8). We use the Cartesian coordinate systems to represent the motions. In order to have human-like motions, the learnt model should be able to generate trajectories with both similar position and velocity profile to the demonstrations. In this experiment, the task was shown to the robot six times, and was learnt using \( K = 6 \) Gaussian functions. Fig. 8(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.

\( \text{Adaptation to Perturbation:} \) Fig. 8(b) shows the robustness of the model to external perturbations. 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 \( t = 1.5 \) second. Having defined the motion as Dynamical Systems, the adaptation to the new target’s position can be done instantly.

\( \text{Note that in our experiments the difference between the two algorithms in terms of the number of parameters is small, and thus is not a decisive factor.} \)

\( \text{The robot is only taught how to move blocks. The problem of grasping the blocks is out of the scope of this paper. Throughout the experiments, we pose the blocks such that they can be easily grasped by the robot.} \)
Increasing Accuracy of Generalization: While convergence to the target is always ensured from conditions given by Eq. 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. 9-Top, 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 given by the teacher is incomplete, and thus the inference for points 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 not covered before. By showing the robot more demonstrations and re-training the model with the new data, the robot is able to successfully accomplish the task (see Fig. 9-Bottom).

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 4 times and were learnt using $K = 4$ Gaussians. The experiments and the generalization of the tasks starting from different points in the space are shown in Fig. 10 and 11. Fig. 12 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 (the saucer for the cup task and the convenience we superimpose both tasks in the same graph. For example, consider Fig. 9-Top, i.e. when the robot starts the motion close to its left fore-hand. The robot is able to successfully follow the motion in front of its face. Then it does a semi-spiral 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 fore-hand. Then it does a semi-circle motion upward and finally brings its arm completely down (see Fig. 14). The two experiments were learnt using 5 and 4 Gaussian functions, respectively. In both experiments the robot is able to successfully follow the demonstrations and to generalize the motion for several trajectories with different starting points. Similarly to what was observed in the three experiments with the Katana robot, the models obtained for the iCub’s experiments are robust to perturbations.

C. Learning 2nd 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. Though this class of ODE functions are generic enough to represent a wide variety of robotic 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 such kinds of motions is the ambiguity in the correct direction of velocity at the intersection point if the model’s variable $\xi$ 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 example, this problem can be solved if one defines the motion in terms of position, velocity, and acceleration, i.e. a 2nd order dynamics:

$$\ddot{x} = g(x, \dot{x})$$  \hspace{1cm} (25)

where $g$ is an arbitrary function. Observe that any second order
11. The Katana-T arm performing the experiment of putting a cup on a tray.

12. The ability of the model to on-the-fly adapt its trajectory to a change in the target’s position.

13. The first experiment with the iCub. The robot does a semi-spiral motion toward its right-side, and at the bottom of the spiral it stretches forward its hand completely.

14. The second experiment with the iCub. The robot does a semi-circle motion upward and brings its arm completely down.
dynamics in the form of Eq. 25 can be easily transformed into a first-order ODE through a change of variable, i.e.:

\[
\begin{align*}
\dot{x} &= v \\
\dot{v} &= g(x, v) \\
\Rightarrow [\dot{x}; \dot{v}] &= f(x, v)
\end{align*}
\] (26)

Having defined \( \xi = [x; v] \) and thus \( \dot{\xi} = [\dot{x}; \dot{v}] \), Eq. 26 reduces to \( \dot{\xi} = f(\xi) \), and therefore can be learnt with the methods presented in this paper. We verify the performance of both methods in learning a 2nd order motion via a robotic 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. 15). Here the task is shown to the robot 5 times. The motion is learnt with seven Gaussian functions with SEDS-Likelihood. The results demonstrate the ability of SEDS to learn 2nd order dynamics. By extension, since any n-th order autonomous ODE can be transformed into a first-order autonomous ODE, the proposed methods can be used to learn arbitrary n-th order dynamics. This is specially helpful if it is desired to model a motion based on other physical properties like jerk, etc.

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. 9). Now we show a more complex example and use SEDS-Likelihood to integrate different motions into one single model (see Fig. 16). In this experiment, the task is learnt using \( K = 7 \) Gaussian functions, and the 2D demonstrations are collected from pen input using a Tablet-PC. The model is learnt 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. 16 we see that all the three dynamics are learnt 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 to Alternative Methods

The proposed method is also compared to three of the best performing regression methods to date (GPR, GMR with EM and LWPR\(^\text{12}\)) and our previous work BM on the same library of handwriting motions represented in Section VI-A (see Table II) and the robotic experiments described in Sections VI-B to VI-D (see Table III). Fig. 3 illustrate the difference between these five methods on the estimation of a 2D 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 non-parametric 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 less 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 joint-wise partitions and ensures that the motion is stable locally.

\(^{12}\)The source code of all the three is downloaded from the website of their Authors.
within each partition. SEDS is more constraining since it tries to fit a motion with a single globally stable dynamics. Finally, in contrast to 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 for learning arbitrary discrete motions by modeling them as nonlinear autonomous DS. We proposed a method called SEDS to learn the parameters of a GMM by solving an optimization problem under strict stability constraint. We proposed two objective functions SEDS-MSE and SEDS-Likelihood for this optimization problem. 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, since when computing the likelihood, the temporal order of datapoints is not important, the objective function can be computed in one-shot. Computing MSE is more time consuming since it requires to integrate the error over the course of the whole trajectory. 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 non-convex objective function. However, in practice, in the 20 handwriting examples and the six robotic tasks, we reported here, we found that SEDS approximation was quite accurate, reaching a close to global optimum. An assumption made throughout this paper is that represented motions can be modeled with a first order time-invariant ODE. While the nonlinear function given by Eq. 9 is able to model a wide variety of motions, the method cannot be used for some special cases violating this assumption. Most of the time, this limitation can be tackled through a change of variable (as presented in our experiments, see Fig. 15).

The stability conditions at the basis of SEDS are sufficient conditions to ensure global asymptotic stability of non-linear 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, local approaches such as Binary Merging (BM) can be used 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, 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 based on the new training points or build a new model from the new demonstrations and merge it with the
previous model. For a fixed number of Gaussians, the former usually results in having a more accurate model, while the latter is faster to train.

Ongoing work is directed at designing an on-line learning version of SEDS whereby the algorithms 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 we followed in [39].

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

Dynamical systems 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 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 work, we presented two learning methods 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 a learning method 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 robotic tasks 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, ability to generalize motions to unseen contexts, and ability to adapt on-the-fly to spatial and temporal perturbations.

ACKNOWLEDGMENT

This work was supported by the European Commission through the EU Projects First-MM (FP7-0248258) and AMARSI (FP7-ICT-248311). The authors kindly thank Jan Peters for his insightful comments during the development of this work, and Eric Sauser for preparing the experimental setup with the iCub robot.

REFERENCES


S.M. Khansari-Zadeh is PhD student in Robotics at the school of Engineering at EPFL. He received his MSc (2008) in Flight Dynamics and Control from department of Aerospace Engineering at Sharif University of Technology. 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 Aude Billard is Associate Professor and head of the Learning Algorithms and Systems Laboratory at the school of Engineering at EPFL. She received her BSc (1994) and MSc (1995) in Physics from EPFL and her PhD in Artificial Intelligence (1998) from the University of Edinburgh.