Learning Control Lyapunov Function to Ensure Stability of Dynamical System-based Robot Reaching Motions

S. Mohammad Khansari-Zadeh*, Aude Billard[1]

EPFL-STI-IMT-LASA, CH-1015, Lausanne, Switzerland

Abstract

We consider an imitation learning approach to model robot point-to-point (also known as discrete or reaching) movements with a set of autonomous Dynamical Systems (DS). Each DS model codes a behavior (such as reaching for a cup and swinging a golf club) at the kinematic level. An estimate of these DS models are usually obtained from a set of demonstrations of the task. When modeling robot discrete motions with DS, ensuring stability of the learned DS is a key requirement to provide a useful policy. In this paper we propose an imitation learning approach that exploits the power of Control Lyapunov Function (CLF) control scheme to ensure global asymptotic stability of nonlinear DS. Given a set of demonstrations of a task, our approach proceeds in three steps: 1) Learning a valid Lyapunov function from the demonstrations by solving a constrained optimization problem, 2) Using one of the-state-of-the-art regression techniques to model an (unstable) estimate of the motion from the demonstrations, and 3) Using (1) to ensure stability of (2) during the task execution via solving a constrained convex optimization problem. The proposed approach allows learning a larger set of robot motions compared to existing methods that are based on quadratic Lyapunov functions. Additionally, by using the CLF formalism, the problem of ensuring stability of DS motions becomes independent from the choice of regression method. Hence it allows the user to adopt the most appropriate technique based on the requirements of the task at hand without compromising stability. We evaluate our approach both in simulation and on the 7 degrees of freedom Barrett WAM arm.

Keywords: Robot point-to-point Movements, Imitation Learning, Control Lyapunov Function, Nonlinear dynamical systems, Stability Analysis, Movement Primitives

1. Introduction

When designing robots meant to interact in a human environment, it is essential to develop methods that can easily transfer skills from nonexpert users to robots, and at the same time, provide the required robustness and reactivity to interact with a dynamic environment. Classical approaches to modeling robot motions rely on decomposing the task execution into two separate processes: *planning* and *execution* [1]. The former is used as a means to generate a feasible path that can satisfy the task's requirements, and the latter is designed so that it follows the generated feasible path as closely as possible. Hence these approaches consider any deviation from the desired path (due to perturbations or changes in environment) as the tracking error, and various control theories have been developed to efficiently suppress this error in terms of some objective functions. Despite the great success of these approaches in providing powerful robotic systems, particularly in factories, they are ill-suited for robotic systems that are aimed to work in the close vicinity of humans, and thus alternative techniques must be sought.

In robotics, Dynamical Systems (DS) based approaches to motion generation have been shown to be interesting alternatives to classical methods as they offer a natural means to integrate planning and execution into one single unit [2, 3, 4, 5]. For instance when modeling robot reaching motions with DS, all possible solutions to reach the target are embedded into one single model [6]. Such a model represents a global map which specifies *instantly* the correct direction for reaching the target, considering the current state of the robot, the target, and all the other objects in the robot's working space. Such models are more similar to human movements in that they can effortlessly adapt its motion to changes in the environment rather than stubbornly following the previous path [6, 7, 8, 9, 10, 11, 12]. In other words, the main advantage of using DS-based formulation can be summarized as: "Modeling movements with DS allows having robotic systems that have inherent adaptivity to changes in a dynamic environment, and that can swiftly adopt a new path to reach the target". This advantage is the direct outcome of having a unified planning and execution unit.

Imitation learning (also known as learning from demonstrations) is one of the most common and intuitive ways of building an estimate of DS motions from a set of demonstrations [6, 13, 7, 8]. When modeling robot motions with DS, *ensuring stability* of the estimated DS – ensuring the robot reaches the final desired state – is one of the main challenges that should

*Ecole Polytechnique Federale de Lausanne, LASA Laboratory, Switzerland.
Email addresses: mohammad.khansari@epfl.ch (S. Mohammad Khansari-Zadeh), aude.billard@epfl.ch (A. Billard)
URL: http://lasa.epfl.ch/
[1]Ecole Polytechnique Federale de Lausanne, LASA Laboratory, Switzerland.

be addressed in order to obtain a useful control policy. This is by construction a difficult conundrum since one needs to deal with the problem of both estimating a nonlinear function, and additionally ensuring stability of an unknown nonlinear DS. Most imitation learning approaches tackle the stability problem by using a time (phase)-dependent clock that smoothly switches from an unstable nonlinear DS to a globally asymptotically stable linear system [7, 8, 12]. However due to the time-dependency, these approaches are sensitive to perturbations. In our previous work [6], we have presented a formal stability analysis to ensure global asymptotic stability of autonomous (i.e. time-invariant) DS. In order to derive stability conditions, that work relied on two assumptions: 1) Robot motions are formulated with Gaussian Mixture Regression (GMR) [14], and more importantly 2) The energy of the motion takes a quadratic form. The former constraints the user to using solely a specific regression method (namely GMR), while the latter limits the range of possible motions that can be modeled accurately.

There are numerous nonlinear regression techniques to estimate nonlinear DS. Each of these techniques has its own pros and cons which make their use very task-dependent. For instance, Gaussian Process Regression (GPR) [15] is an accurate method but is computationally expensive. GMR is computationally fast, but is comparatively less accurate, for instance, than GPR. Locally Weighted Projection Regression (LWPR) [16] is a powerful tool for incremental learning but requires setting several initial parameters. Various advantages and disadvantages can also be observed when using other techniques such as Support Vector Regression (SVR) [17], Reservoir Computing (RC) [18], and Gaussian Process Latent Variable Model (GPLVM) [19]. Thus, it would be advantageous if one could freely choose the most appropriate regression technique based on requirements of the task at hand, while still ensuring the robot can reach the desired target point. Note that the standard training of the above regression techniques do not ensure global asymptotic stability at the target [6].

The field of control theory has provided us with various tools to design a stable controller for (nonlinear) DS around a desired target point. Control Lyapunov Function (CLF) control scheme [20, 21, 22] is one of these techniques that is designed based on the following intuitive idea: "Associate a Lyapunov function (i.e. energy function) $V(\boldsymbol{\xi})$ with its global minimum at the target point $\boldsymbol{\xi}^*$ to the DS $\boldsymbol{f}(\boldsymbol{\xi})$ that needs to be stabilized. At each time step, apply a stabilizing (or control) command $\boldsymbol{u}(\boldsymbol{\xi})$ so as to force $V(\boldsymbol{\xi})$ decreases. This system is guaranteed to asymptotically reach the target point."

In this paper we propose an imitation learning approach that exploits the power of Control Lyapunov Function (CLF) control scheme to ensure global asymptotic stability of DS-based robot reaching motions. For brevity, we call the proposed approach CLF-DM (Control Lyapunov Function-based Dynamic Movements). Given a set of demonstrations of a task, our approach proceeds in three steps: 1) Learning a valid Lyapunov function from a set of demonstrations, 2) Using one of the-state-of-the-art regression techniques to model an (unstable) estimate of the motion from the demonstrations, and 3) Using (1) to ensure stability of (2) at the unique target point during the task execution. The proposed approach allows learning a larger set of robot motions compared to existing methods that are based on quadratic energy function. Additionally, due to the CLF formalism, the problem of ensuring stability of DS motions becomes independent from the choice of regression method. Thus, one could now adopt the most appropriate technique based on the requirements of the task at hand without compromising stability. Furthermore, with the new approach, one has the possibility to have online/incremental learning which is crucial in many tasks as it allows the user to refine the robot motion in an interactive manner.

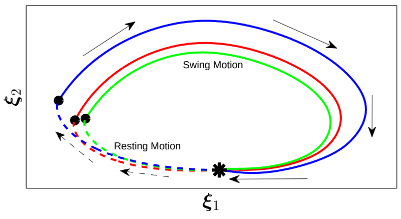
In contrast to classical CLF-based approaches where the energy function is mostly hand-tuned by the user, a non-trivial problem that has only been solved for special classes of DS [23], in our approach we build an estimate of the CLF from demonstrations. Additionally, by learning CLF, our approach is tailored to generate stabilizing commands that modify the unstable DS given by $\boldsymbol{f}(\boldsymbol{\xi})$ as least as possible at each iteration. Hence, it tries to maximize the similarity between the stabilized and the unstable DS which is crucial to generate motions that resemble the user demonstrations. Note that apart from the CLF approach, the Optimal Control techniques [24] can also be exploited so as to generate a sequence of stabilizing commands to ensure stability of $\boldsymbol{f}(\boldsymbol{\xi})$. Despite the successful realtime implementation of these approaches for linear DS, their implementation for nonlinear DS is still an open question and realtime solutions only exist for particular cases.

The contribution of this work are four-folds: 1) Proposing a new parameterization for energy function, called Weighted Sum of Asymmetric Quadratic Function (WSAQF), that significantly outperforms the classical way of modeling energy function with a quadratic function, 2) Presenting a constrained optimization problem to build an estimate of a task-oriented Lyapunov function from a set of demonstrations, 3) Proposing an optimal control problem (based on the learned Lyapunov function) to ensure stability of nonlinear autonomous DS, and 4) Extending the classical CLF control scheme and present it in the context of learning robot discrete motions from demonstrations. By taking this approach, we provide the user with the possibility to choose a regression technique of her choice (for example based on the requirements of the task at hand). We evaluate the proposed approach on a library of human handwriting motions and on the 7 degrees of freedom Barrett WAM arm.

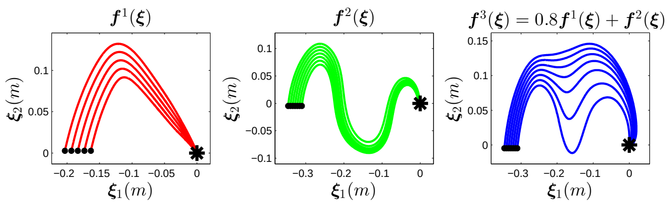
2. Related Work

The problem of motion generation for robot movement has been an active research topic in robotics for years, and many techniques have been suggested addressing different aspects of this problem. In this section we only focus on reviewing Dynamical System (DS) based approaches as it is the main topic covered in this work[2]. The DS approach to modeling robot mo-

[2]Interested readers could refer to [25] for more information about the difference between DS-based approaches and other techniques such as path planners [26], time-indexed trajectory generators [27], and potential field methods [28].



(a) This graph shows an example of a tennis swing, which is composed of a swing and a resting phase. The motion in each phase is encoded as a basic movement primitive, and the complete tennis swing motion is thus obtained by sequencing these two primitives. The trajectories generated by the swing and resting models are shown in solid and dashed lines, respectively. The direction of the motion is indicated by arrows. Only three examples of generated trajectories are shown here (plotted as red, green, and blue lines).



(b) In this example $f^1(\xi)$ and $f^2(\xi)$ are two basic movement primitives that represent an angle and sine-shaped motion, respectively. The new movement primitive $f^3(\xi)$ that includes a mixture of both behaviors is obtained through a linear superposition of $f^1(\xi)$ and $f^2(\xi)$.

Figure 1: Illustration of two examples exploiting modularity of DS models to generate **(a)** a more complex motion and **(b)** a new movement primitive. In this figure, the black star and circles indicate the target and initial points, respectively.

tions is a type of *feedback motion planning*. Hence a controller driven by a stable DS is robust to perturbations because it embeds all possible solutions to reach a target into one single function. During the last decade, DS has been used to model discrete motions [6, 7, 29, 30], rhythmic motions [31, 32], hitting motions [33, 34, 7], obstacle avoidance [35, 36, 37], etc.

In DS approaches, the control policy to drive a robotic platform is modeled with a first or higher order DS. When controlled through a DS, a robot motion unfolds in time with no need to re-plan. An estimate of the DS can be built from a few demonstrations of the task at hand. The estimated DS captures the invariant features in the user demonstrations, and can generate motions that resemble the user demonstrations [13]. Each DS model codes a specific motion (behavior), and is called a movement primitive (also known as motor primitive). They can be seen as a building block that can be used to generate more complex or new motions through sequencing or superimposition of the primitives. This modularity of DS-based movement primitives is essential as it allows controlling a wide repertoire of movements from a (small) set of basic motions [38, 39]. Figure 1 shows two examples of exploiting this modularity of movement primitives to generate new motions[3].

The idea of using a programmable DS formulation to generate motions is not new and has been an active topic for decades. The Vector Integration To Endpoint (VITE) model is one of the early approaches that is suggested to simulate arm reaching movements [40, 41]. Central Pattern Generators (CPGs) are also another type of DS model that are suggested to model rhythmic behaviors [42, 43].

Recurrent Neural Network (RNN) and its variants [44, 45, 46] are another type of DS-based tools that have been used to model discrete and rhythmic motions (besides to its application in other domains such as signal processing, vision systems, and system identification). In general terms, RNNs are computational models that are composed of numerous interconnected neurons. As it appears from its name, connection topology in RNN includes cycles which allows to render it to be a DS. An overview of different RNN structures and training algorithms are presented in [47].

In the context of robot imitation learning, Schaal et. al. were among the first groups to suggest the idea of using a programmable DS formulation that can be adjusted to different tasks [3]. This idea was then further extended in [48], where they proposed a method, called Dynamic Movement Primitives (DMP), to build an estimate of nonlinear DS via Imitation Learning. DMP offers a method by which a nonlinear DS can be estimated while ensuring global stability at the attractor. DMP has been used for different robotics applications such as: walking [32], drumming [30], pouring [49], and obstacle avoidance [36, 37].

Along with the idea of modeling robot motions with autonomous (i.e. time-invariant) DS, in [50] we proposed an alternative approach, called Binary Merging (BM), to construct a *locally* stable estimate of nonlinear autonomous DS as a mixture of Gaussian functions. Though this work provided sufficient conditions to make DS *locally* stable, it relied on determining numerically the stability region.

In [29, 51], Calinon et. al. suggested a DS approach based on Hidden Semi-Markov Model and Gaussian Mixture Regression (GMR). In these works, sole a brief verification to avoid large instabilities was suggested; however, asymptotic stability could not be ensured.

In our previous work, Stable Estimator of Dynamical systems (SEDS) [6], we developed for the first time a *formal* stability analysis and formulated explicit constraints to ensure global asymptotic stability of nonlinear autonomous DS that are modeled as a mixture of Gaussian functions. This work has been further extended to perform striking movements [52, 33], obstacle avoidance [35], infinite Gaussian mixture model using a Dirichlet process [9], hand-arm coordination [53], stiffness control [54], catching flying objects [55], and multi-attractors DS [56].

The rest of this paper is structured as follows: Section 3 describes our revised DS formulation to generate robot motions. Section 4 formalizes the problem of learning a Control Lyapunov Function (CLF) from a set of demonstrations. Section 5 provides the formulation to ensure stability of DS-based discrete robot motions based on the learned CLF. Section 6 presents the experimental results. Section 7 discusses the assumptions and limitations of this work, and Section 8 concludes the paper.

[3]Remark that nonlinear sum of two or more stable DS is not necessarily stable, and special attention should be considered in this regard (see [6] for further discussion).

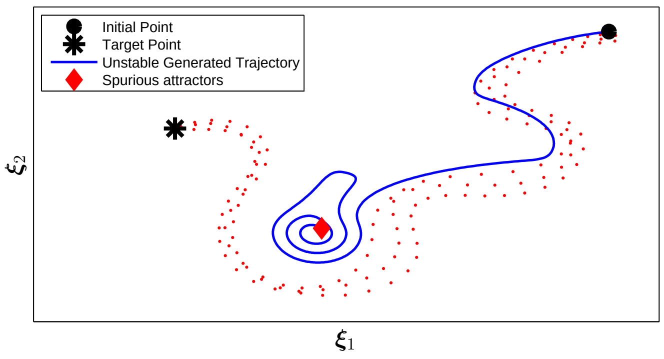


Figure 2: Illustration of convergence to a spurious attractor when estimating a two-dimensional dynamics on the basis of three training examples (shown with red dots) using standard GMM. This undesired behavior is mainly because standard regression techniques do not consider asymptotic stability of DS during their training.

3. Dynamical Systems-Based Robot Motions

We formulate robot discrete motions as a control law driven by autonomous Dynamical Systems (DS). Consider a state variable $\boldsymbol{\xi} \in \mathbb{R}^d$ that can be used to unambiguously define a discrete motion of a robotic system (e.g. $\boldsymbol{\xi}$ could be a robot's joint angles, position and/or orientation of an arm's end-effector in the operational space, etc):

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}), \quad \boldsymbol{f} : \mathbb{R}^d \mapsto \mathbb{R}^d \tag{1}$$

where $\boldsymbol{f}(\boldsymbol{\xi})$ is a continuous function that codes a specific behavior such as reaching for a cup and swinging a golf club. Starting in an initial configuration $\boldsymbol{\xi}^0$, the robot motion $\boldsymbol{\xi}^t, t \in [0\ \infty)$ is given by integrating from Eq. (1). As outlined before, we consider the class of motions which ends to a single point $\boldsymbol{\xi}^*$, i.e. the attractor of the system. Typical examples of such motion are reaching out for an object, closing fingers in a particular grasping configuration, stepping motion, etc. We hence seek to derive an estimate of $\boldsymbol{f}(\boldsymbol{\xi})$ that is globally asymptotically stable at this attractor so as to ensure that, even when perturbed, the system will ultimately reach the goal. Furthermore, we seek to have an estimate that follows a particular dynamics (shown by the user through a set of demonstrations) throughout the state space.

We will start by building an estimate of $\boldsymbol{f}(\boldsymbol{\xi})$ based on a set of N demonstrations $\mathcal{D} = \{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\}_{t=0,n=1}^{T^n,N}$ using any of the state-of-the-art regression methods. We will denote the estimated DS with $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$. In our experiments, the training points are provided by a human demonstrator, passively guiding the robot through the motion (to avoid the correspondence problem). This initial estimate of $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ will be likely unstable or converge to a spurious attractor because standard regression techniques do not consider asymptotic stability of DS during their training. Figure 2 shows an example of such unstable behavior when estimating $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ from three demonstrations using the standard GMR with Expectation-Maximization training algorithm.

As discussed before, in this paper we present a CLF-based approach to ensure all motions reach the unique attractor $\boldsymbol{\xi}^*$. In this approach, we first need to associate a Lyapunov function to the DS at hand. Let us consider again the example presented in Fig. 2 and assume the energy of the motion is defined by a function $V(\boldsymbol{\xi})$. Note that throughout this paper we use the terms 'Lyapunov function' and 'Energy function' interchangeably. Figure 3 illustrates the energy levels of $V(\boldsymbol{\xi})$. Following our example, at the initial point the system has the energy $V(\boldsymbol{\xi}^0)$. As the trajectory evolves by integrating from $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$, its energy continuously dissipates until it reaches a point where the rate of change in energy becomes zero (indicated with a red triangle). We call this point a *divergence point* because the energy of the system will start increasing after this point by following $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ (see the blue dashed line in Fig. 3) and eventually converge to a spurious attractor.

In order to avoid this behavior, following the CLF principal, the estimate from $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ should be corrected by applying a stabilizing command $\boldsymbol{u}(\boldsymbol{\xi}, \hat{\boldsymbol{f}}(\boldsymbol{\xi}))$ so as to force the energy of the motion to keep decreasing. Our control system is hence composed of two terms, an initial estimate of the nonlinear dynamics given by $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ and a secondary stabilizing terms $\boldsymbol{u}(\boldsymbol{\xi})$, i.e.:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) = \hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi}) \tag{2}$$

The stabilizing command $\boldsymbol{u}(\boldsymbol{\xi})$ must provide corrections such that $\hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi})$ will have a component along the negative direction of the gradient of $V(\boldsymbol{\xi})$ except at the target point, i.e.

$$\begin{aligned}
&\text{Determine } \ \boldsymbol{u}(\boldsymbol{\xi}) \in \mathbb{R}^d \\
&\text{such that:} \\
&\quad (\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}))^T \left(\hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi})\right) < 0 \qquad \forall \boldsymbol{\xi} \in \mathbb{R}^d \setminus \boldsymbol{\xi}^* \\
&\quad (\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}))^T \left(\hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi})\right) = 0 \qquad \boldsymbol{\xi} = \boldsymbol{\xi}^*
\end{aligned} \tag{3}$$

where $(.)^T$ denotes the transpose. The stabilizing command $\boldsymbol{u}(\boldsymbol{\xi})$ is a multi-dimensional vector, and thus there are usually more than one solution to Eq. (3) (there are d variables and only one constraint). In Section 5, we will discuss how to determine an optimal choice for $\boldsymbol{u}(\boldsymbol{\xi})$. For the purpose of this example, we can simply assume that we choose any of the solutions to Eq. (3), so as to drive the motion in the direction that decreases the energy. The region that requires such corrections is indicated by a thick blue line in Fig. 3. At the end of this region, the estimate solely given by $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ is enough to derive the motion at the direction that decreases the energy until it reaches a region close to the target, where it requires again some corrections and then finally reaches the target point. Following this control scheme, we could ensure global asymptotic stability of $\boldsymbol{f}(\boldsymbol{\xi})$ at the unique target point $\boldsymbol{\xi}^*$ independently from how the unstable estimate $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ is obtained.

Figure 4 shows the schematic of the control flow proposed in this paper. First, note the difference between the stabilizing command $\boldsymbol{u}(\boldsymbol{\xi})$ and the robot control command $\boldsymbol{\tau}$. The former is a virtual signal that is generated at the kinematic level to make the DS planer stable, while the latter is the actual torque/force that is applied to the robot and is computed using an inverse dynamics controller in order to follow the desired motion. A

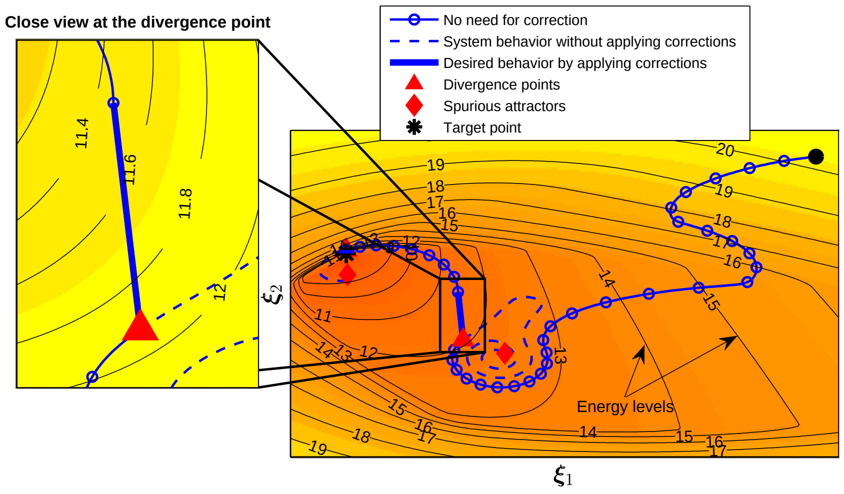


Figure 3: Illustration of the principal idea behind the presented approach. In this example, the trained DS is unstable and converges to two spurious attractors (indicated by red diamonds). In order to stabilize this system, we first associate an energy (Lyapunov) function to the system. Recall that the energy function is positive everywhere except at the target where it vanishes. In this graph, the energy function is depicted with color gradient: the lighter the color, the higher the energy. At each time, we verify if the motion is moving at the direction that decreases the energy. If this is the case, then we do not need to change the motion. Otherwise, we correct the estimated DS so as to move at the direction that decreases the energy. By doing so, the system is guaranteed to be globally asymptotically stable at its unique target point.

forward/inverse kinematic block may also be required if the motion is defined in the operational space. In this control architecture, the stability of the DS motion generator is ensured while executing the task based on the CLF. In the motion encoding block, an estimate of DS model is constructed based on the demonstrations using the user preferred regression model. Depending on the regression method, the motion encoding step may be performed offline or online. In this paper, we only consider an offline approach to learn CLF.

4. Learning CLF from demonstrations

As outlined in the previous section, in order to ensure global stability of $\boldsymbol{f}(\boldsymbol{\xi})$ at the target $\boldsymbol{\xi}^*$, we need to determine a Lyapunov function $V(\boldsymbol{\xi})$ for the DS at hand. The proper estimation of $V(\boldsymbol{\xi})$ plays an important role on the amount of corrections that will be applied to the system through $\boldsymbol{u}(\boldsymbol{\xi})$, and subsequently on the accuracy with which $\boldsymbol{f}(\boldsymbol{\xi})$ can follow the user demonstrations.

Figure 5 illustrates the importance of having a proper energy function on a 2D toy example. As one can see, the regions that require stabilization (indicated by colored patches) can significantly change by choosing different Control Lyapunov Functions (CLF). In this example, the energy function that is shown in Fig. 5c seems more ideal than the other two cases due to the following reason: it applies almost zero correction on the region that is covered by the user demonstrations. This is a crucial feature as it yields a stabilized DS $\boldsymbol{f}(\boldsymbol{\xi})$ that can accurately follow the demonstrations, or in other words the energy function is consistent with the demonstrations.

4.1. Approach

To have an energy function that is consistent with the provided demonstrations, we take an imitation learning approach and propose a new learning algorithm to build an estimate of the energy function from the demonstrations.

Definition 1 *We call the function $V(\boldsymbol{\xi})$* consistent with the user demonstrations *if 1) V is positive $\forall \boldsymbol{\xi} \in \mathbb{R}^d \setminus \boldsymbol{\xi}^*$, 2) It has a unique global minimum at the target point $\boldsymbol{\xi}^*$, where $V(\boldsymbol{\xi}^*) = 0$, and 3) As one moves along any of the demonstration trajectories, V decreases as time increases (i.e. $V(\boldsymbol{\xi}^{t,n}) > V(\boldsymbol{\xi}^{t+1,n})$) and finally vanishes at the end of the demonstration, $V(\boldsymbol{\xi}^{T^n,n}) = 0$.*

The first two requirements in Definition 1 are standard conditions to ensure V is a valid Lyapunov function. They also naturally lead the vector of partial derivatives to vanish at the target, i.e. $\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}^*) = 0$. The third criterion is essential to ensure no correction will be applied at regions covered by the demonstrations (see Fig. 5). This condition is equivalent to requesting that $\dot{V}$ is negative for all training data points except the final point (i.e. target), and can be verified by computing the scalar product between the energy gradient and the velocity at each training data point, i.e. for each $\{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\}_{t=0,n=1}^{T^n-1,N}$ require:

$$\begin{aligned}\dot{V}(\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}) &= \frac{dV(\boldsymbol{\xi}^{t,n})}{dt} = \left(\frac{dV(\boldsymbol{\xi}^{t,n})}{d\boldsymbol{\xi}}\right)^T \frac{d\boldsymbol{\xi}^{t,n}}{dt} \\ &= \left(\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}^{t,n})\right)^T \dot{\boldsymbol{\xi}}^{t,n} < 0 \end{aligned} \tag{4}$$

In this paper, we take an optimization approach and build an estimate of the energy function from the user demonstrations by solving a constrained optimization problem. We formulate the first two requirements in Definition 1 as the constraints of the optimization problem. We define the optimization objective function so that it maximizes the number of training data points that satisfy Eq. (4), i.e. the third requirement. The more the training data points that satisfy Eq. (4), the better the estimated energy function represents the demonstrations. It should be noted that the conditions given by Eq. (4) cannot be considered in the optimization constraints as it might not be possible to ensure them for all the training data points. This is partially due to the fact that the demonstrations are noisy observations of instances of the DS, and partially due to the way in which the energy function is parameterized. The former is inevitable as some of the data points may even contradict each other by having, for example, different velocity profiles at the same point. The latter directly controls the amount of nonlinearity that can

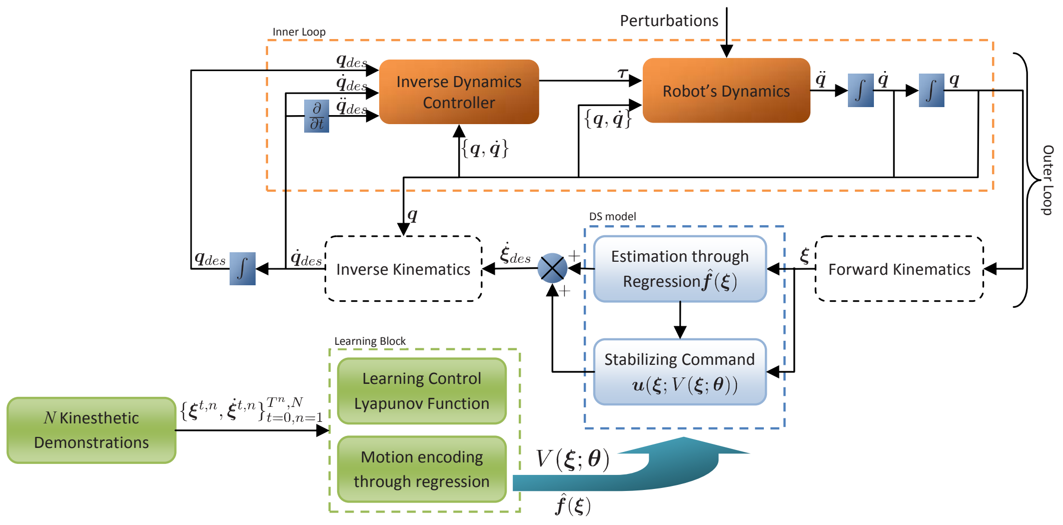
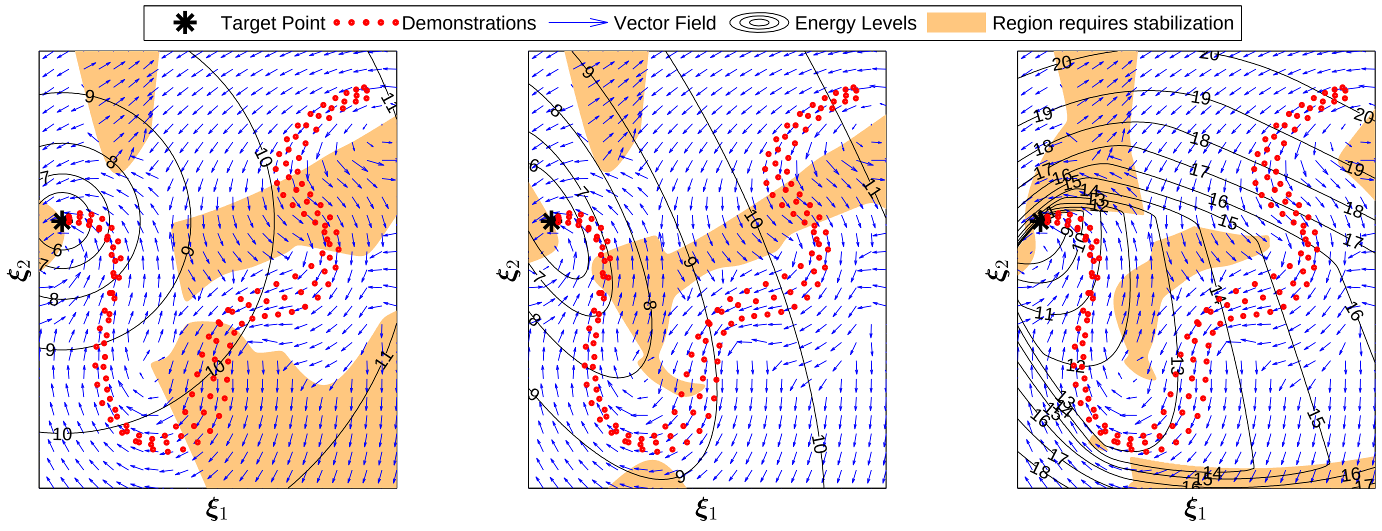

Inner Loop
Perturbations
Inverse Dynamics Controller
Robot's Dynamics
Outer Loop
Inverse Kinematics
DS model
Estimation through Regression $\hat{f}(\xi)$
Forward Kinematics
Stabilizing Command $u(\xi; V(\xi;\theta))$
Learning Block
Learning Control Lyapunov Function
Motion encoding through regression
N Kinesthetic Demonstrations
$\{\xi^{t,n}, \dot{\xi}^{t,n}\}_{t=0,n=1}^{T^n,N}$
$V(\xi;\theta)$
$\hat{f}(\xi)$
q_{des}
$\dot{q}_{des}$
$\ddot{q}_{des}$
τ
$\ddot{q}$
$\dot{q}$
q
$\{q,\dot{q}\}$
$\dot{\xi}_{des}$
ξ

Figure 4: The system's architecture illustrating the control flow when using the presented approach to derive the stable control policy. In this graph, $\boldsymbol{q}$, $\boldsymbol{\tau}$, and $\boldsymbol{\xi}$ correspond to the robot's joint angles, joint torques, and the state variables describing the robot motion, respectively.



(a) First candidate for $V(\boldsymbol{\xi})$ **(b)** Second candidate for $V(\boldsymbol{\xi})$ **(c)** Third candidate for $V(\boldsymbol{\xi})$

Figure 5: Illustration of the importance of having a proper energy function on a 2D toy example through three different candidates for the energy function. As one can see, the regions that require stabilization can significantly change by choosing different CLF. Note that the DS $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ is the same in all the three cases.

be captured with V, and thus its effect can be reduced or eliminated by choosing a more complex parameterization for the energy function[4]. We will elaborate more about the parameterization of V in Section 4.2.

4.2. Parameterization

To obtain an estimate of $V(\boldsymbol{\xi})$ from the demonstrations, we first proceed by parameterizing it with a vector of parameters $\boldsymbol{\theta}$ and denote it with $V(\boldsymbol{\xi};\boldsymbol{\theta})$. In this paper we propose a new parameterization, called Weighted Sum of Asymmetric Quadratic Functions (WSAQF). As it appears from its name, the WSAQF parameterization models the energy function as a weighted sum of asymmetric quadratic functions:

$$V(\boldsymbol{\xi};\boldsymbol{\theta}) = (\boldsymbol{\xi}-\boldsymbol{\xi}^*)^T \boldsymbol{P}^0 (\boldsymbol{\xi}-\boldsymbol{\xi}^*) + \sum_{\ell=1}^{\mathscr{L}} \beta^\ell(\boldsymbol{\xi};\boldsymbol{\theta}) \Big((\boldsymbol{\xi}-\boldsymbol{\xi}^*)^T \boldsymbol{P}^\ell (\boldsymbol{\xi}-\boldsymbol{\mu}^\ell-\boldsymbol{\xi}^*) \Big)^2 \tag{5}$$

where $\mathscr{L}$ is the number of asymmetric quadratic functions defined by the user, $\boldsymbol{\mu}^\ell \in \mathbb{R}^d$ are vectors influencing the asymmetric shape of the energy function, $\boldsymbol{P}^\ell \in \mathbb{R}^{d\times d}$ are positive definite matrices, and the coefficients $\beta^\ell(\boldsymbol{\xi};\boldsymbol{\theta})$ are:

$$\beta^\ell(\boldsymbol{\xi};\boldsymbol{\theta}) = \begin{cases} 1 & \forall \boldsymbol{\xi} : (\boldsymbol{\xi}-\boldsymbol{\xi}^*)^T \boldsymbol{P}^\ell (\boldsymbol{\xi}-\boldsymbol{\mu}^\ell-\boldsymbol{\xi}^*) \geq 0 \\ 0 & \forall \boldsymbol{\xi} : (\boldsymbol{\xi}-\boldsymbol{\xi}^*)^T \boldsymbol{P}^\ell (\boldsymbol{\xi}-\boldsymbol{\mu}^\ell-\boldsymbol{\xi}^*) < 0 \end{cases} \tag{6}$$

Note that due to the way the energy function $V(\boldsymbol{\xi};\boldsymbol{\theta})$ is defined, $V(\boldsymbol{\xi};\boldsymbol{\theta})$ is continuous and has continuous first order partial derivatives (i.e. class $\mathcal{C}^1$ smoothness) even when $\beta^\ell(\boldsymbol{\xi};\boldsymbol{\theta})$ switches value from 1 to 0 and vice-versa. Furthermore, although the value of $V(\boldsymbol{\xi};\boldsymbol{\theta})$ remains positive for any arbitrary values of $\boldsymbol{P}^\ell, \ell = 1..\mathcal{L}$, the requirement on their positive defi-

[4]As for analogy, consider the example of fitting a polynomial on a set of data points that are sampled from a nonlinear function. In this example, the choice of the order of polynomials directly affect the accuracy with which the underlying function is estimated.

niteness is essential to ensure that $V(\boldsymbol{\xi};\boldsymbol{\theta})$ has a unique global minimum, see the second requirement in Definition 1.

There are three main advantages in using the WSAQF parameterization: 1) It allows us to bypass the first two criteria in Definition 1 just by requiring the matrices $\boldsymbol{P}^\ell$ to be positive definite. This verification can be done easily, hence significantly reducing the workload of the learning algorithm. As a result, compared to numerical approaches for stability evaluation, the WSAQF parameterization can better scale up to higher dimensional dataset. 2) By construction, it ensures $\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}) \neq \mathbf{0}, \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^*$ and $\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}^*) = \mathbf{0}$. An important feature that makes the energy function free from local minima and the target point the unique global minimum of the energy function. 3) This form of energy function allows having energy components that act locally without introducing any discontinuity in the total energy of the system. As a result, a considerably wider set of motions can be modeled compared to the conventional quadratic energy function.

4.3. Learning Algorithm

The learning parameters of WSAQF are the components of $\boldsymbol{P}^\ell$ and $\boldsymbol{\mu}^\ell$, i.e. $\boldsymbol{\theta} = \{\boldsymbol{P}^0, ..., \boldsymbol{P}^{\mathscr{L}}, \boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^{\mathscr{L}}\}$. A locally optimal solution to $\boldsymbol{\theta}$, and thus equivalently the energy function, can be found by solving the following constrained optimization problem:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \qquad \sum_{n=1}^{N}\sum_{t=0}^{T^n} \frac{(1+\bar{w})\text{sign}(\psi^{t,n})}{2}(\psi^{t,n})^2 + \cdots$$
$$+ \frac{(1-\bar{w})}{2}(\psi^{t,n})^2$$
$$\text{subject to} \tag{7}$$
$$\boldsymbol{P}^\ell \succ 0 \qquad \forall \ell = 0..\mathscr{L}$$

where $\succ$ denotes the positive definiteness of a matrix, $\bar{w}$ is a small positive scalar (i.e. $\bar{w} > 0$ and $\bar{w} \ll 1$), and $\psi^{t,n}$ is:

$$\psi^{t,n} = \psi(\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}; \boldsymbol{\theta}) = \frac{(\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}^{t,n};\boldsymbol{\theta}))^T \dot{\boldsymbol{\xi}}^{t,n}}{\|\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}^{t,n};\boldsymbol{\theta})\|\|\dot{\boldsymbol{\xi}}^{t,n}\|} \tag{8}$$

Throughout this paper we use the interior-point algorithm to solve this optimization problem [57]. Note that by defining the objective function as described above, the solver first favors lowering the number of data points for which Eq. (4) does not hold, and as a second priority, it tries to align the gradient of energy with the negative direction of movement. By tuning the value of $\bar{w}$, one can control the priority portion of these two objectives. Furthermore, the normalization by the norm of the gradient and velocity vectors in Eq. (8) is essential to give an equal importance to all training data points during the optimization. Note that, the global asymptotic stability of $\boldsymbol{f}(\boldsymbol{\xi})$ can be ensured irrespective of the number of data points for which Eq. (4) does not hold. However, the lower the number of data points that violate Eq. (4), the lesser the amount of corrections applied to $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ in the region(s) covered by the demonstrations.

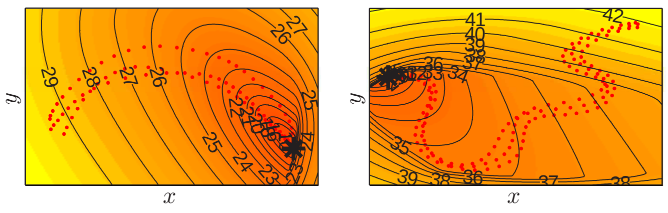


Figure 6: Two examples of estimating the energy function from a set of user demonstrations using the WSAQF parameterization with $\mathscr{L} = 1$ (left) and $\mathscr{L} = 3$ (right). In this figure, we consider $\boldsymbol{\xi} = [x; y]$. The background color also illustrates the energy level. The lighter the color, the higher the energy.

Figure 6 shows two examples of energy functions that are learned from a set of user demonstrations using the proposed approach. Here the first example is a single-curve motion, and the second one is a multi-curve motion. An estimate of the energy function for these motions are obtained using $\mathscr{L} = 1$ and $\mathscr{L} = 3$, respectively. In these motions, all the training data points satisfy Eq. (4).

The proper choice of the number of asymmetric functions is important. The higher the $\mathscr{L}$, the more complex the energy function that can be modeled. However, the increase in complexity comes at the cost of having more optimization parameters. Furthermore by using a large number of asymmetric functions, one may overfit the nonlinearities due to the noise in the demonstrations. Thus, there is a compromise inherent in setting the value of $\mathscr{L}$. In the experiments we present here, we proceed as follows: we start with $\mathscr{L} = 0$, i.e. the conventional Quadratic Energy Function (QEF) parameterization. Then we incrementally increase $\mathscr{L}$ until it no longer affects or marginally improves the objective function in Eq. (7). In all our experiments, an energy function with $\mathscr{L} \leq 4$ resulted in a proper energy parameterization.

5. Computation of Stabilizing Command

In Section 4 we proposed an optimization problem to compute a valid energy function based on the user demonstrations. Now it remains to determine a proper stabilizing command so as to force the motion at the direction that decreases the energy function. As discussed before, there are usually infinite solutions to this problem. We could use this redundancy to choose a $\boldsymbol{u}(\boldsymbol{\xi})$ that is optimal with respect to task-oriented criteria.

For the problem at hand, we are interested in modifying the estimated Dynamical System (DS) $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ as least as possible within a finite or infinite horizon. Hence, we use this metric to choose the best stabilizing command among the possible solutions by solving a constrained optimization problem. In this paper, we take a greedy approach and try to find the minimal value for $\boldsymbol{u}$ at each time step such that $\hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi})$ has a component along $-\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi})$ except at the target point, i.e.:

$$\min_{\boldsymbol{u}(\boldsymbol{\xi})} \qquad \frac{1}{2}\boldsymbol{u}(\boldsymbol{\xi})^T\boldsymbol{u}(\boldsymbol{\xi})$$
$$\text{subject to:} \tag{9}$$
$$(\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}))^T \left(\hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi})\right) \leq -\rho(\|\boldsymbol{\xi}\|) \qquad \forall \boldsymbol{\xi} \in \mathbb{R}^d \setminus \boldsymbol{\xi}^*$$
$$\hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \boldsymbol{u}(\boldsymbol{\xi}) = 0 \qquad \boldsymbol{\xi} = \boldsymbol{\xi}^*$$

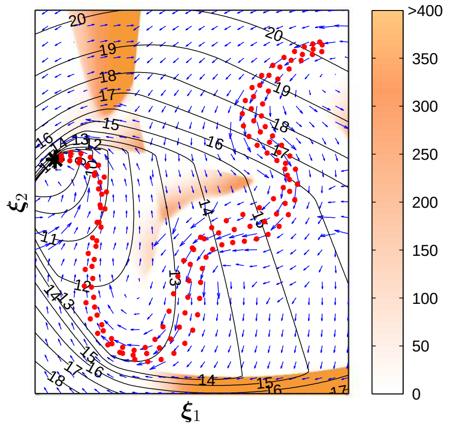


Figure 7: Applying $\boldsymbol{u}^*(\boldsymbol{\xi})$ from Eq. (10) to ensure global asymptotic stability of $\boldsymbol{f}(\boldsymbol{\xi})$ for the 2D toy example of Fig. 5. The value of $\frac{1}{2}\boldsymbol{u}^*(\boldsymbol{\xi})^T\boldsymbol{u}^*(\boldsymbol{\xi})$ is indicated with color gradient (unit is mm^2/s^2).

where $\rho(\|\boldsymbol{\xi}\|)$ belongs to class $\mathcal{K}$ function and imposes the minimum acceptable rate of decrease in the energy function[5]. Note that the use of $\rho(\|\boldsymbol{\xi}\|)$ is necessary to convert the strict inequality constraint in Eq. (3), which is not suitable for optimization, into an inequality constraint by redefining the constraint upper bound. In this paper we consider $\rho(\|\boldsymbol{\xi}\|) = \rho_0(1 - e^{-\kappa_0\|\boldsymbol{\xi}\|})$ with $\rho_0 > 0$ and $\kappa_0 > 0$.

The optimization problem given by Eq. (9) is convex and has a unique global minimum, which can be determined analytically:

$$\boldsymbol{u}^*(\boldsymbol{\xi}) = \begin{cases} \mathbf{0} & \forall \boldsymbol{\xi} \in \mathbb{R}^d \setminus \boldsymbol{\xi}^*, \nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi})^T \hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \rho(\|\boldsymbol{\xi}\|) \le 0 \\ -\hat{\boldsymbol{f}}(\boldsymbol{\xi}) & \boldsymbol{\xi} = \boldsymbol{\xi}^* \\ -\left(\frac{\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi})^T \hat{\boldsymbol{f}}(\boldsymbol{\xi}) + \rho(\|\boldsymbol{\xi}\|)}{\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi})^T \nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi})}\right) \nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}) & \text{elsewise} \end{cases} \tag{10}$$

The optimal stabilizing command $\boldsymbol{u}^*(\boldsymbol{\xi})$ can be always found given a Lyapunov function $V(\boldsymbol{\xi})$ as per Section 4 (note that the WSAQF parameterization by construction ensures $\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}) \neq \mathbf{0}, \forall \boldsymbol{\xi} \neq \boldsymbol{\xi}^*$). Hence global asymptotic stability of $\boldsymbol{f}(\boldsymbol{\xi})$ can be always guaranteed at its unique target point $\boldsymbol{\xi}^*$ independently from the type of regression technique that is used to estimate $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$. However, the more accurate the $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$, the lesser effort $\boldsymbol{u}^*(\boldsymbol{\xi})$ is required for the DS stabilization. Figure 7 shows the stabilization of DS by applying the optimal $\boldsymbol{u}^*$ for the example shown in Fig. 5.

The complete procedure about how to use the presented approach is summarized in Algorithm 1. As described before, our approach proceeds in three steps: (Step-1) Learning a valid Lyapunov function from a set of demonstrations by solving a constrained optimization problem, (Step-2) Using one of the-state-of-the-art regression techniques to model an (unstable) estimate of the motion from the demonstrations, and (Step-3) Using the learned energy function from Step-1 to ensure stability of the DS model obtained in Step-2 during the task execution via solving a constrained convex optimization problem. The first and third steps were described in detail in Sections 4 and 5, respectively. The second step is composed of two sub-steps: (a) correcting the dataset according to the learned energy function, and (b) using the corrected dataset to build an estimate of the motion $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ using the regression technique of our choice. Although the sub-step (a) does not play any role in ensuring stability of $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$, it is essential for faithful training of the model.

To elaborate more, consider a demonstration data point $\{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\} \in \mathcal{D}$ that is unstable according to the learned energy function $V(\boldsymbol{\xi};\boldsymbol{\theta}^*)$, i.e. $\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}^{t,n};\boldsymbol{\theta}^*)^T\dot{\boldsymbol{\xi}}^{t,n} \ge 0$. If this data point is learned accurately, i.e. $\dot{\boldsymbol{\xi}}^{t,n} = \hat{\boldsymbol{f}}(\boldsymbol{\xi}^{t,n})$, then it is necessary to correct the velocity command during motion execution at this point and some region around it. Thus, ensuring accurate estimate at this data point is undesirable as this would require to apply a correction later on with $\boldsymbol{u}^* \neq \mathbf{0}$. To avoid encountering this issue, we could first correct all demonstration data points before learning the estimate $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ to ensure that all demonstration data points yield a consistent decrease of $V(\boldsymbol{\xi};\boldsymbol{\theta}^*)$, and thus preventing $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ to learn a behavior that will be ignored during motion execution in Step-3. We next explain how we proceed to these corrections on the demonstration data points. Let us denote the corrected dataset as $\hat{\mathcal{D}} = \{\boldsymbol{\xi}^{t,n}, \hat{\dot{\boldsymbol{\xi}}}^{t,n}\}_{t=0,n=1}^{T^n,N}$, where the corrected velocities $\hat{\dot{\boldsymbol{\xi}}}^{t,n}$ are computed according to:

$$\hat{\dot{\boldsymbol{\xi}}}^{t,n} = \dot{\boldsymbol{\xi}}^{t,n} + \boldsymbol{u}^*(\boldsymbol{\xi}^{t,n}) \qquad \forall\{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\} \in \mathcal{D} \tag{11}$$

Note that the correction of the demonstrations is only done on the velocity part. Furthermore, $\boldsymbol{u}^*(\boldsymbol{\xi}^{t,n}) = \mathbf{0}$ if the data point $\{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\} \in \mathcal{D}$ is stable according to $V(\boldsymbol{\xi};\boldsymbol{\theta}^*)$. Considering our example above, now all the training data points satisfy $\nabla_{\boldsymbol{\xi}} V(\boldsymbol{\xi}^{t,n};\boldsymbol{\theta}^*)^T\hat{\dot{\boldsymbol{\xi}}}^{t,n} < 0$, $\forall t < T^n$. Therefore, training $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ on $\hat{\mathcal{D}}$ requires less correction during the motion execution in region close to the demonstrations compared to the other case. We would like to highlight that correcting the demonstration data points is optional. However, regardless of whether the dataset is corrected or not, it is still necessary to apply stabilizing command during motion execution in Step-3 to ensure global asymptotic stability of $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ at the target.

6. Experiments

We evaluate the performance of the proposed approach in three ways: 1) On one complex planar motion that is inferred from human demonstrations. With this experiment, we illustrate one of the main properties of the proposed method, namely that it can be used to stabilize unstable DS that are modeled with different regression techniques, 2) In a robot experiment performed on the 7-DoF Barrett WAM arm that requires online/incremental learning. In this experiment, we demonstrate that our approach allows combining two different regression techniques in order to benefit from the advantages of both, and 3) Through comparison with our previous approach Stable Estimator of Dynamical Systems (SEDS) [6]. This comparison is evaluated based on a library of 2D human handwriting motions.

[5]A scalar continuous function $\rho(r)$, defined for $r \in [0, a)$ is said to belong to class $\mathcal{K}$ function if it is strictly increasing and $\rho(0) = 0$ [58].

Algorithm 1 Control Lyapunov Function-based Dynamic Movements (CLF-DM)

```
Step-1: Learning the Lyapunov Function:
Input: $\mathcal{D} = \{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\}_{t=0,n=1}^{T^n,N}$ and a threshold value $\kappa$
 1: $\mathscr{L} \leftarrow 0$
 2: $i \leftarrow 0$
 3: while true do
 4:     Estimate $V(\boldsymbol{\xi};\boldsymbol{\theta}^i)$ by solving Eq. (7).
 5:     if $i > 0$ and $\|J(\boldsymbol{\theta}^i) - J(\boldsymbol{\theta}^{i-1})\| < \kappa$ then
 6:         break
 7:     end if
 8:     $\mathscr{L} \leftarrow \mathscr{L} + 1$
 9: end while
10: $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}^{i-1}$
Output: $V(\boldsymbol{\xi};\boldsymbol{\theta}^*)$

Step-2: Learning the (unstable) DS Model:
Input: $\mathcal{D} = \{\boldsymbol{\xi}^{t,n}, \dot{\boldsymbol{\xi}}^{t,n}\}_{t=0,n=1}^{T^n,N}$ and $V(\boldsymbol{\xi};\boldsymbol{\theta}^*)$
11: (Optional): Correct $\mathcal{D}$ based on $V(\boldsymbol{\xi};\boldsymbol{\theta}^*)$ using Eq. (11) and save
    it as $\hat{\mathcal{D}}$.
12: Learn $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ from $\hat{\mathcal{D}}$ using a regression method.
Output: $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$

Step-3: Robot Execution:
Input: $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$, $V(\boldsymbol{\xi};\boldsymbol{\theta}^*)$, $\rho(\|\boldsymbol{\xi}\|)$, initial state $\boldsymbol{\xi}^0$, target state $\boldsymbol{\xi}^*$, accu-
    racy tolerance $\epsilon$, and integration time step $\delta t$
13: $i \leftarrow 0$
14: while $\|\boldsymbol{\xi}^i - \boldsymbol{\xi}^*\| > \epsilon$ do
15:     Compute $\hat{\boldsymbol{f}}(\boldsymbol{\xi}^i)$
16:     Compute $\boldsymbol{u}^*(\boldsymbol{\xi}^i)$ from Eq. (10)
17:     $\dot{\boldsymbol{\xi}}^i \leftarrow \hat{\boldsymbol{f}}(\boldsymbol{\xi}^i) + \boldsymbol{u}^*(\boldsymbol{\xi}^i)$
18:     Compute next state $\boldsymbol{\xi}^{i+1} \leftarrow \boldsymbol{\xi}^i + \dot{\boldsymbol{\xi}}^i\delta t$
19:     $i \leftarrow i + 1$
20: end while
```

6.1. *Learning a Complex Motion with four Regression Methods*

In this experiment, demonstrations of the motion are collected at 50Hz from pen input using a Tablet-PC. This motion is complex in that it is composed of trajectories that begin by distancing themselves from the target point, and then approach it (see Fig. 8a). Thus, a conventional quadratic function cannot represent the energy flow of this motion.

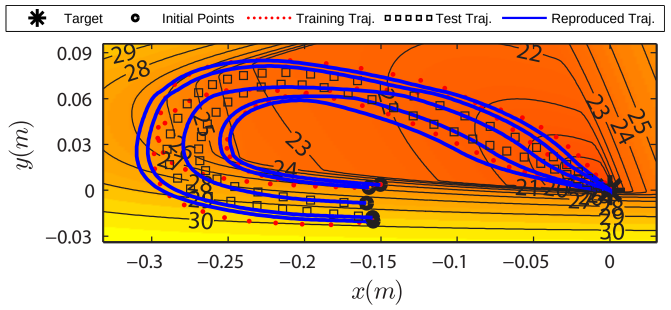
As described in Section 4, we first start by building an estimate of the energy function from the demonstrations. For this motion, we model the energy function with $\mathscr{L} = 3$. Then we build an unstable estimate of $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ using four different regression techniques including epsilon Support Vector Regression (ϵ-SVR), Gaussian Mixture Regression (GMR), Gaussian Process Regression (GPR), and Locally Weighted Projection Regression (LWPR). As the estimation of the energy function is independent from the regression method, we use the same energy function to generate the stabilizing command for all the four regression techniques[6]. Furthermore, this feature also allows comparing these four techniques and then choosing the one that fits the task best. In this section we evaluate these approaches based on the Correction Effort (CE); however, one could also use other criteria depending on the requirements of the task. The CE can be measured by computing the total applied stabilizing command divided by the total traveled length that these corrections amount to:

$$CE = \frac{\sum_{t=0}^{tf} \|\boldsymbol{u}(\boldsymbol{\xi}^t, \hat{\boldsymbol{f}}(\boldsymbol{\xi});\boldsymbol{\theta})\|\delta t}{\sum_{t=0}^{tf} \|\dot{\boldsymbol{\xi}}^t\|\delta t} = \frac{\sum_{t=0}^{tf} \|\boldsymbol{u}(\boldsymbol{\xi}^t, \hat{\boldsymbol{f}}(\boldsymbol{\xi});\boldsymbol{\theta})\|}{\sum_{t=0}^{tf} \|\dot{\boldsymbol{\xi}}^t\|} \tag{12}$$

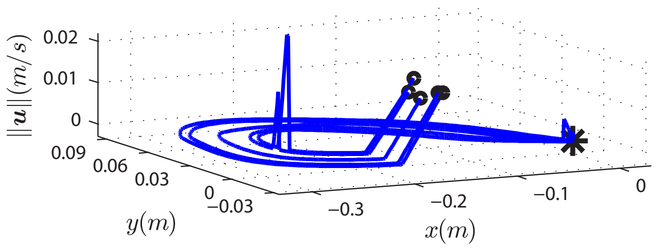
where δt is the integration time step. The lower CE, the more stable behavior $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ has along that trajectory, and thus the less it is distorted. If $CE = 0$, it means that $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ is naturally stable starting in the given initial point.

The quantitative comparison between these techniques for this motion is summarized in Table 1. The results are computed on a set of two test trajectories (indicated with black squares in Fig. 8). For this motion, the best result in terms of the average stabilizing command is obtained with GMR, although the other three methods demonstrate a comparative performance. The trajectories obtained using the GMR model are shown in Fig. 8a.

The norm of stabilizing commands for this motion is illustrated in Fig. 8b. As we can see, only in small parts of the trajectories, the stabilizing command is active in order to ensure



(a) Modeling an unstable estimate of the motion with GMR, and learning its energy function using $\mathscr{L} = 3$. The background color illustrates the energy level. The lighter the color, the higher the energy.



(b) Illustration of the norm of the stabilizing command.

Figure 8: Performance evaluation of the proposed algorithm in learning a complex planar motion. Here, we consider $\boldsymbol{\xi} = [x; y]$.

Table 1: Performance comparison between ϵ-SVR, GMR, GPR, and LWPR in learning a complex planar motion shown in Fig. 8. Obtained values are averaged across all the demonstrations.

Criterion	ϵ-SVR	GMR	GPR	LWPR
CE	1.32%	**0.35%**	1.99%	2.54%

[6]The following values are used for $\rho(\|\boldsymbol{\xi}\|)$: $\rho_0 = 0.1$ and $\kappa_0 = 0.001$.

convergence to the target. On average for the GMR model, the value of the correction effort is 0.35%. Furthermore, the activation of the stabilizing command on a small region around the target verifies that $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ is unstable, and without using the presented approach, all trajectories would have missed the target. Finally, one can observe that the generated trajectories resemble the user demonstrations despite applying stabilizing commands. This is mainly due to the fact that stabilizing commands are derived from an energy function that is estimated based on the user demonstrations. Note that the result from this experiment does not necessarily imply that GMR outperforms other regression techniques for any kind of motions. Instead, it aims at providing a simple means to choose the best regression technique for the task at hand.

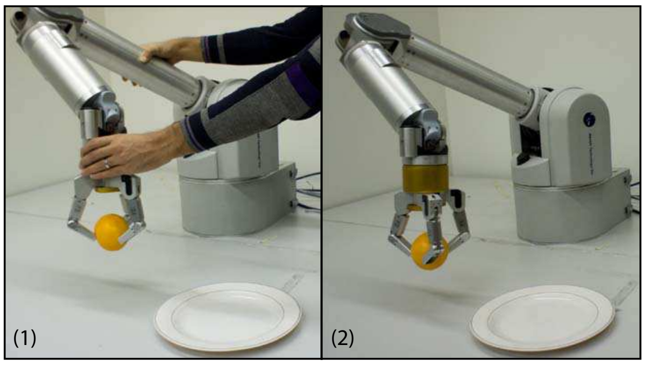
6.2. Robot Experiments

The robot experiment consisted of having the WAM arm place an orange on a plate and into a bucket. First, the placing task on the plate is shown to the robot seven times via kinesthetic teaching (see Fig. 9a). A DS estimate of this motion is constructed in the Cartesian space, i.e. $\boldsymbol{\xi} = [x; y; z]$, using GMM with 7 Gaussian functions. The energy function is also estimated based on the user demonstrations using the Weighted Sum of Asymmetric Quadratic Function (WSAQF) parameterization with $\mathcal{L} = 0$. The demonstrations and the reproductions of the task from the proposed method are shown in Fig. 9b. As it is illustrated, the reproductions closely follow the demonstrations, while their global stability is ensured. Note that the robot is controlled at 500Hz, which indicates the overall computation time of the proposed approach is less than 2 millisecond.

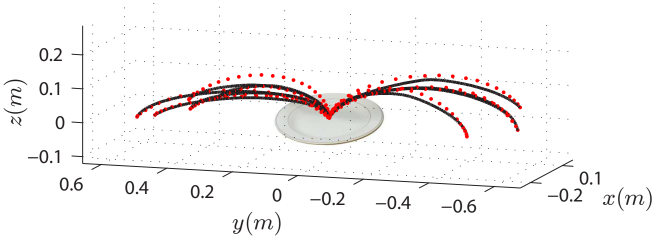
In the second part of this experiment, we replace the plate with a bucket. Due to the specific shape of the bucket, the previously learned model is no longer suitable for the new situation (see the solid black lines in Fig. 10a). In order to adapt to this change without retraining the whole model, we exploit the online learning power of LWPR to locally modify the DS given by GMR:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}) = \underbrace{\boldsymbol{g}(\boldsymbol{\xi}) + \boldsymbol{l}(\boldsymbol{\xi})}_{\hat{\boldsymbol{f}}(\boldsymbol{\xi})} + \boldsymbol{u}(\boldsymbol{\xi}) \tag{13}$$

where $\boldsymbol{g}(\boldsymbol{\xi})$ and $\boldsymbol{l}(\boldsymbol{\xi})$ correspond to the DS that are modeled with GMR and LWPR, respectively. Initially $\boldsymbol{l}(\boldsymbol{\xi}) = 0, \forall \boldsymbol{\xi} \in \mathbb{R}^d$. The LWPR model is trained incrementally and online by interactively correcting the robot movement while it approaches the bucket (see Fig. 10b). The blue hollow circles in Fig. 10a shows the new training data points that were collected interactively as we have explained. Figure 10c illustrates the reproductions from the combined DS according to Eq. (13). The stabilizing command is generated using the same energy function as before. With the new model, the robot can successfully adapt its motion and place the orange into the bucket. Note that in this experiment, the GMR model grants the base behavior for the placing task, and the LWPR model provides the required adaptation to the environment. Anytime when it is necessary, the base behavior can be retrieved by canceling out the LWPR term in Eq. (13). By extension, one can also imagine having several



(a) The demonstration of the task through kinesthetic teaching (left), and its execution by the robot (right).



(b) Generation of trajectories from different initial points

Figure 9: The robot experiment of placing an orange on a plate. The placing motion is modeled with GMR.

LWPR models, each of which provides the required adaptive behavior for different containers.

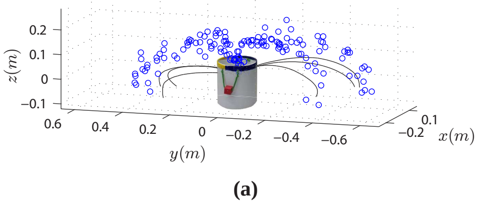
6.3. Comparison to Our Previous Approach

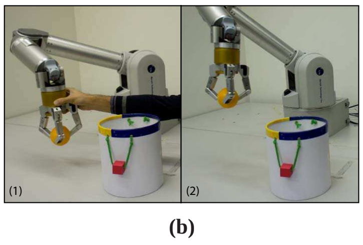
In this section we compare the presented approach to our previous work Stable Estimator of Dynamical Systems (SEDS) [6]. This comparison is made based on a library of 2D human handwriting motions. We use SEDS with likelihood as the objective function, and consider CLF-DM in combination with the unstable estimates from GPR, GMR, LWPR, and SVR. For each motion, the evaluation is made on a set of six test trajectories that are spread in between and outside the training trajectories (see Fig. 11). All reproductions were generated in simulation to exclude the error due to the robot controller from the modeling error. The Quadratic Energy Function (QEF) parameterization is also used to encode the energy function of all the 20 motions in the library.

The qualitative comparison of the estimate of the handwriting motions are provided in Fig. 12. Quantitative performance comparisons in terms of the accuracy in estimation and the training time are summarized in Fig. 13. We use the swept error area as a metric to evaluate the estimation accuracy of each method which is computed by:

$$\mathscr{E} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T^n}\mathcal{A}(\boldsymbol{\xi}^n(t), \boldsymbol{\xi}^n(t+1), \boldsymbol{\xi}^{t,n}, \boldsymbol{\xi}^{t+1,n}) \tag{14}$$

where $\mathcal{A}(\boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3, \boldsymbol{p}^4)$ corresponds to the area of the tetragon generated by the four points $\boldsymbol{p}^1$ to $\boldsymbol{p}^4$, $\boldsymbol{\xi}^n(t) = \sum_{i=0}^{t}\dot{\boldsymbol{\xi}}^n(i)dt$ generate an estimate of the corresponding demonstrated trajectory $\boldsymbol{\xi}^n$ by starting from the same initial points as those demonstrated, i.e. $\boldsymbol{\xi}^n(0) = \boldsymbol{\xi}^{0,n}, \forall n \in 1..N$. Figure 14 shows an





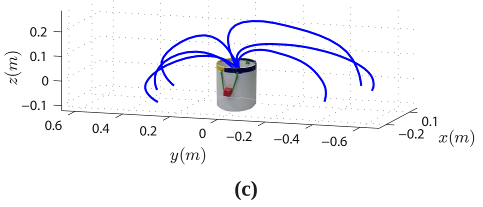


Figure 10: Adaptation of the DS for the case where the plate is replaced with the bucket. **(a)** In this situation, the previous model can no longer be used. To adapt to this change, the user interactively modifies the robot trajectory as the robot approaches the bucket. The data collected during the human interaction is used for online training of the LWPR model, and are shown with blue hollow circle. The solid black lines represent trajectories generated with solely using the GMR model. **(b)** Training interactively the DS model in order to adapt to the new situation (**b**-left). The execution of the reaching motion with the combined LWPR+GMR DS (**b**-Right). **(c)** Trajectories generated with the combined GMR+LWPR DS. With the new model the robot can successfully place the orange into the bucket.

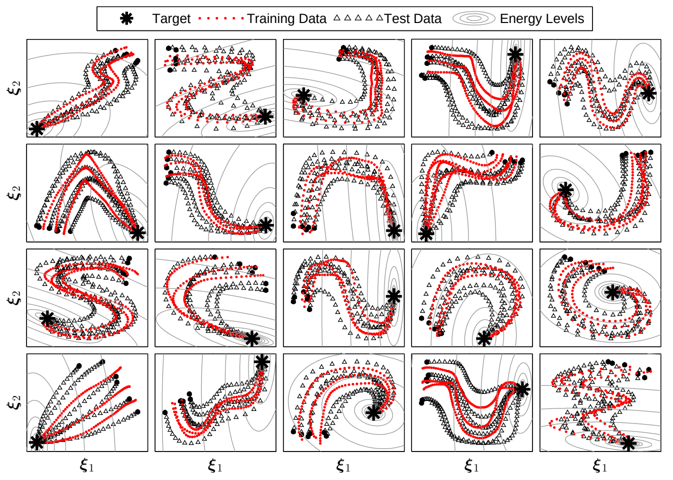


Figure 11: Demonstrations of the training and test trajectories for the 20 non-linear 2D motions that are used for comparison across SEDS and CLF-DM.

example of the swept error area between the reference and the generated trajectory for a 2D motion.

The result on the estimation accuracy are shown in Fig. 13a. As we can see, all the methods are capable of providing the same order of accuracy in estimations. However, the comparison result on the training time indicates that CLF-DM with GMR and SVR formulations are the fastest methods that can build a stable estimate of DS within a few seconds (see Fig. 13b). Note that the training time for the CLF-DM methods includes the energy function learning step. As it is indicated, the average time to learn the energy function is less than a second, which shows the fast computation speed of the proposed learning algorithm. All the methods have a retrieval time (i.e. computing $\dot{\boldsymbol{\xi}}$ for a given $\boldsymbol{\xi}$) of less than a millisecond, which makes them suitable for realtime execution on robots.

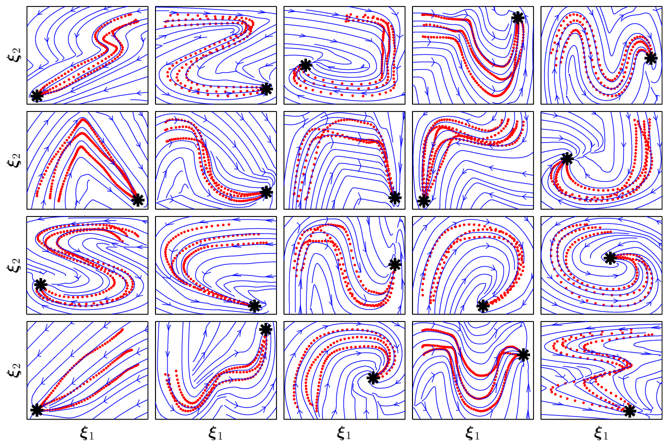
As outlined before, the energy function of all the 20 handwriting motions were encoded using the QEF parameterization. Now let us compare our approaches on a set of more advanced motions where the use of WSAQF parameterization becomes crucial in order to model the energy function more accurately. We conduct this comparison between SEDS and CLF-DM with GMR encoding on a set of five motions (see Fig. 15). Note that for brevity we omit the other possible approaches as the different variants of CLF-DM provide the same order of accuracy.

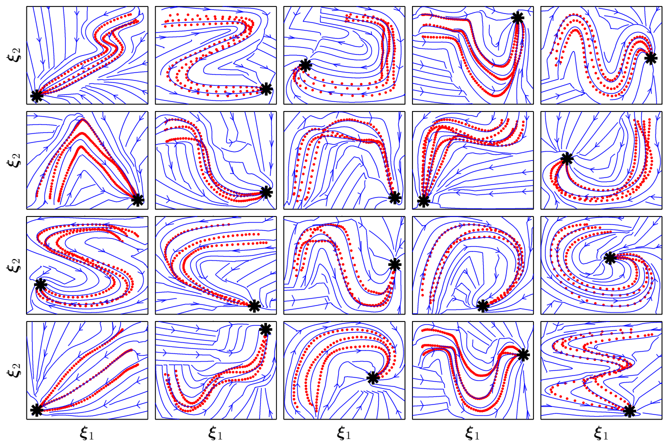
The comparison result is illustrated in Fig. 16. As we can see, while CLF-DM performs equally well, the performance of SEDS is degraded. This is due to fact that the stability conditions in SEDS are derived based on the assumption that $V(\boldsymbol{\xi})$ is modeled with QEF parameterization. As it is illustrated in Fig. 15, the trajectories generated from the SEDS model deviate from demonstrations in order to respect the conditions derived from the quadratic energy function. In contrast, the trajectories from CLF-DM can follow the demonstrations as it allows using more complex energy functions.

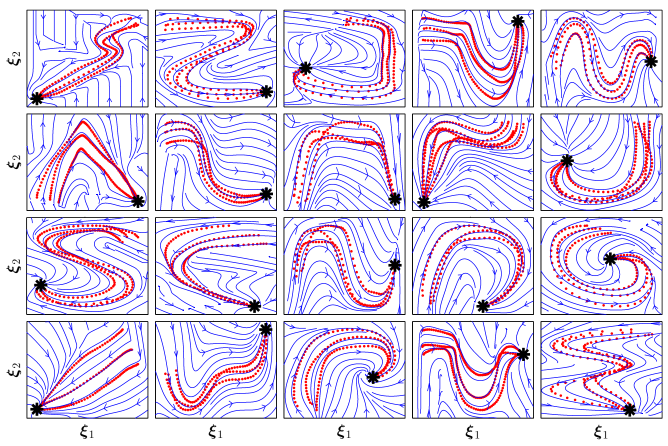
The main advantages and disadvantages of these methods are summarized in Table 2. In most cases, CLF-DM provides the most flexible yet accurate means of building stable DS from demonstrations. In contrast to the SEDS, CLF-DM can model a larger set of tasks and allow using different regression techniques to encode motions. The latter is advantageous especially when online learning is required. The use of SEDS is preferable if 1) the energy of the motion can be represented with the QEF parameterization, and 2) neither online learning nor other regression techniques than GMR is necessary for the task. In these cases, SEDS can (marginally) outperforms CLF-DM. This is mainly because in CLF-DM, learning the energy function and $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ are two separate processes. In contrast, due to the quadratic form that SEDS considers for the energy function, it is possible to unify these two processes into one single training algorithm, hence (possibly) reaching a better local optimum model.

7. Discussion

The algorithm presented in this paper ensures global asymptotic stability of $\boldsymbol{f}(\boldsymbol{\xi})$ at the target, and thus by construction brings the movement to a stop at the target point. The extension of asymptotically stable DS to perform striking movements (i.e. movement with non-zero final velocity) is described in our previous work [33]. In essence, this extension can be achieved by reformulating the robot motion with a target field $\boldsymbol{E}(\boldsymbol{\xi}) : \mathbb{R}^d \mapsto \mathbb{R}^d$ and a strength factor $v(\boldsymbol{\xi}) : \mathbb{R}^d \mapsto \mathbb{R}^+$. The target field is computed based on the asymptotically stable DS, e.g. CLF-DM or SEDS, and defines for each point $\boldsymbol{\xi}$ in state space a normalized vector specifying the direction of motion. The strength factor is a positive scalar, which defines the speed


(a) CLF-DM with GMR encoding.


(b) CLF-DM with LWPR encoding.


(c) CLF-DM with GPR encoding.

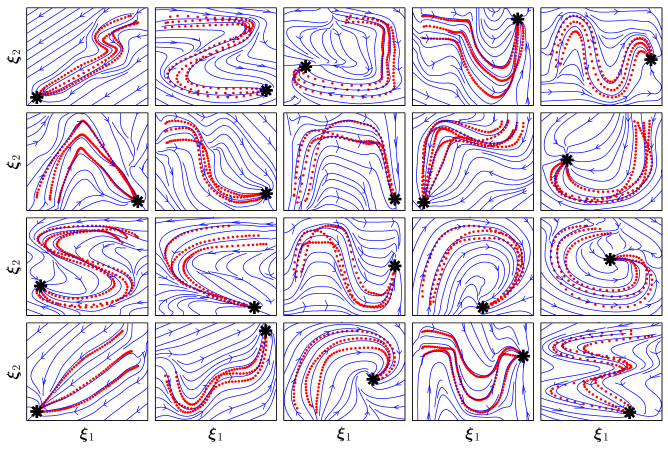

(d) CLF-DM with SVR encoding.

Figure 12: Qualitative performance evaluation on the library of 20 human handwriting motions.

of motion for each point $\boldsymbol{\xi}$. The multiplication of the target field by the strength factor allows the control policy to perform striking movement with hitting speed $v(\boldsymbol{\xi}^*)$ and direction $\boldsymbol{E}(\boldsymbol{\xi}^*)$ at the target point with guaranteed convergence to the target.

In this work, we took a greedy approach and computed the minimum stabilizing command $\boldsymbol{u}^*(\boldsymbol{\xi})$ at each time step. An extension of this approach can be seen in the context of Model Predictive Control (MPC) [59]. In MPC, one optimizes for a finite horizon in the future instead of one single step, and thus could result in a smaller value for the stabilizing command. However, this comes at the cost of solving a constrained nonlinear optimization problem which is computationally more expensive as it should be solved iteratively (no closed-form solution is available). This problem can be alleviated by linearizing both $\hat{\boldsymbol{f}}(\boldsymbol{\xi})$ and $V(\boldsymbol{\xi};\boldsymbol{\theta})$ around the query point $\boldsymbol{\xi}^t$. However special care should be taken as the linearized model is only accurate for a certain neighborhood around $\boldsymbol{\xi}^t$.

The presented approach controls the robot trajectories at the kinematic level. Thus, as outlined before, we assume there is a low level tracking controller that converts kinematic variables into motor commands through an inverse dynamics controller (see Fig. 4). This control scheme is often associated with one main concern: "the hardware limitations of the robot, such as the torque limit, are not considered at the level of trajectory generation". However, contrary to the classical planer, this concern is not very critical when using a DS-based model for trajectory generation. In fact, the DS model can compensate for deviations (due to hardware limitations) from the desired trajectory, by instantly adapting a new trajectory for the new position of the robot. In other words, it treats the robot's hardware limitations similarly to perturbations. It should be noted that an inevitable outcome of such compensation is that the robot executes the motion at a slower pace than what is expected.

All theorems derived in this section are based on the continuous state space assumption; however, in real experiments, robot motions are usually generated with a finite number of points (discrete modeling). Thus the choice of integration time step is important as a big integration time step could cause instability in the system even though the continuous DS model is globally asymptotically stable. However, this should not be such an issue as most of the robotic systems usually operate in a sufficiently high frequency (e.g. the WAM arm operates at 500Hz).

In this work, we proposed a constrained optimization problem to build an estimate of the energy function based on a set

Table 2: Summary of the structural differences between SEDS and CLF-DM.

Method	Type of stability	The level at which stability is ensured	Regression model	Learning mode	Multiple motions encoding	Computational complexity
SEDS	Global	Training	GMR	Offline	Yes	Light
CLF-DM	Global	Execution	Any	Offline/online[†]	Yes	Light

[†] The online learning support depends on the type of regression that is selected.

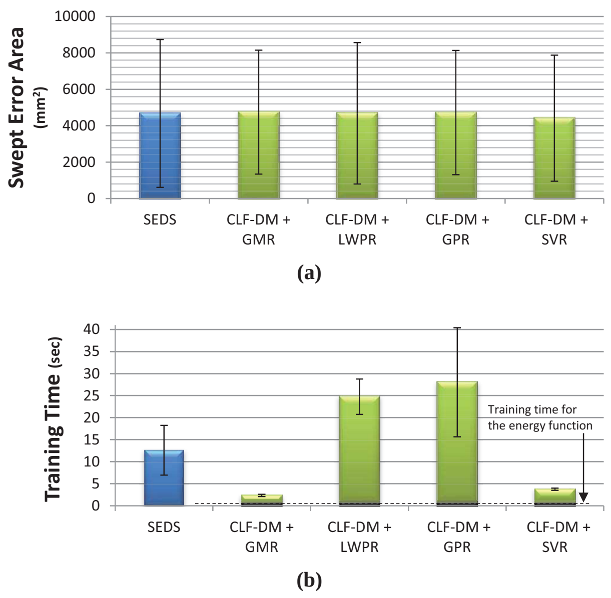


Figure 13: Performance comparison between the presented approach and SEDS on a library consists of 20 human handwriting motions. For clarity, we depict the result from SEDS and CLF-DM with different colors. For further information and discussion please refer to Section 6.3.

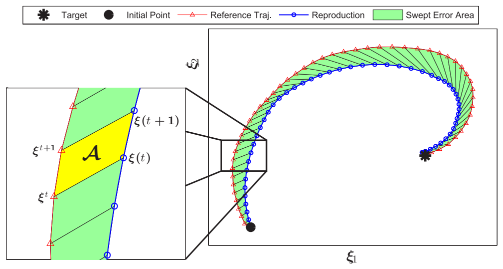
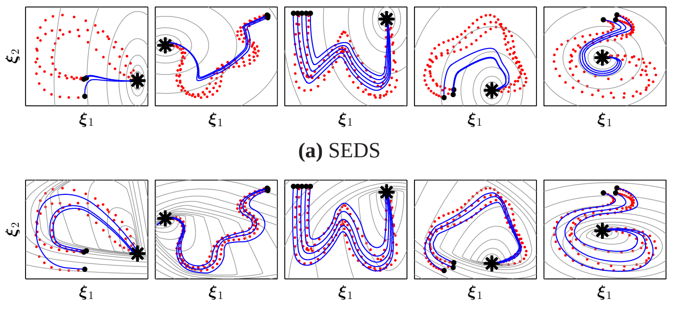


Figure 14: Illustration of the swept error area between the reference and the generated trajectories.



(b) CLF-DM with GMR encoding. From left to right, the WSAQF energy function is modeled with $\mathscr{L} = 3, 1, 1, 2,$ and 1 asymmetric quadratic functions, respectively.

Figure 15: Qualitative comparison between SEDS and CLF-DM on a set of five advanced motions (note that x and y-axes have different scale).

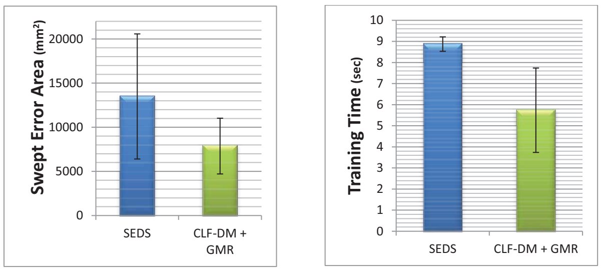


Figure 16: Performance comparison between SEDS and CLF-DM with GMR encoding on a set of five advanced motions, where the use of WSAQF parameterization becomes crucial to accurately model the energy function.

of demonstrations. To solve this optimization problem, the presented approach requires parameterization of the energy function. The choice of parameterization can influence the accuracy of the final result, hence it should be chosen appropriately. The WSAQF parameterization presented in this paper allows estimating the energy function of complex motions while significantly reducing the workload of the optimization problem. However, it also puts an upper bound on the range of energy functions that can be accurately encoded with the WSAQF. In this parameterization the energy of the system should increase by radially moving away from the target. In order to properly model a motion that violates this requirement, one could decompose it into two or more segments and then learn each segment separately. This solution is equivalent to the funnel-based approaches [60] as each funnel takes the motion from one part of the task space and guides it to the basin of attraction of the next funnel until it reaches the target.

Furthermore, as outlined before, online learning is often crucial to allow the user to refine the model in an interactive manner. Our approach allows online learning of DS models through the use of LWPR (or other possible regression techniques that support online learning). However the new updates through online learning might be ignored after applying the stabilizing command. In other words, the result from online learning is only valid if it is consistent with the estimated energy function, which is currently learned offline. In most cases, the above assumption is not critical as there are some flexibilities in generating stabilizing command, which allows the output from the DS to form an angle between $-\pi/2$ and $\pi/2$ with the gradient of the energy function. Thus, as long as the modifications through online learning do not fundamentally change the global features of the motion, it is very likely that the original energy function would not impose any limitation. However, to entirely remove this issue, one could adopt an online training technique to estimate the energy function.

8. Summary and Conclusion

In this paper, we presented a new technique, called Control Lyapunov Function-based Dynamic Movements (CLF-DM), to ensure global asymptotic stability of autonomous multi-dimensional DS. Given a set of demonstrations of a task, CLF-DM proceeds in three steps: 1) Learning a valid Lyapunov function from the demonstrations by solving a constrained optimization problem, 2) Using one of the-state-of-the-art regression techniques to model an (unstable) estimate of the motion from the demonstrations, and 3) Using (1) to ensure stability of (2) during the task execution via solving a constrained convex optimization problem.

To learn the energy function in the first step, we proposed a new parameterization, called Weighted Sum of Asymmetric Quadratic Function (WSAQF), that significantly outperforms the classical way of modeling energy function with quadratic functions. We also presented a learning algorithm to determine a (locally) optimal estimate of the energy function by solving a constrained optimization problem. For the second step, we discussed two different metrics, namely Correction Effort and Swept Error Area, that can be used to help the user choosing the most appropriate regression technique for the task at hand. For the third step, we proposed an optimal control problem to ensure stability of the estimated DS from the previous step. This optimal control problem can be solved analytically (it has a closed from solution) and is thus suited for realtime robot application.

The presented approach was evaluated on a set of theoretical and robot experiments. Compared to existing approaches, CLF-DM is able to learn a larger set of motions because it learns the energy function instead of using a predefined one. Additionally, it allows users to choose the most appropriate regression techniques based on the requirements of the task at hand.

Acknowledgments

This work was supported by the European Commission through the EU Project AMARSi (FP7-ICT-248311).

References

[1] O. Brock, O. Khatib, Elastic Strips: A framework for integrated planning and execution, in: Proceedings of the International Symposium on Experimental Robotics, Vol. 250, Springer Verlag, 1999, pp. 328–338.

[2] J. Kelso, Dynamic patterns: The self-organization of brain and behavior, Cambridge, MA: MIT Press, 1995.

[3] S. Schaal, S. Kotosaka, D. Sternad, Nonlinear dynamical systems as movement primitives, in: proceedings of the IEEE-RAS International Conference on Humanoid Robots, 2000.

[4] A. Billard, G. Hayes, DRAMA, a connectionist architecture for control and learning in autonomous robots, Adaptive Behavior Journal 7 (1) (1999) 35–64.

[5] A. I. Selverston, Are central pattern generators understandable?, The Behavioral and Brain Sciences 3 (1980) 555–571.

[6] S. M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with Gaussian mixture models, IEEE Trans. on Robotics 27 (5) (2011) 943–957.

[7] J. Kober, K. Mulling, O. Kromer, C. H. Lampert, B. Scholkopf, J. Peters, Movement Templates for Learning of Hitting and Batting, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2010, pp. 853–858.

[8] P. Kormushev, S. Calinon, D. G. Caldwell, Robot Motor Skill Coordination with EM-based Reinforcement Learning, in: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), 2010, pp. 3232–3237.

[9] V. Kruger, V. Tikhanoff, L. Natale, G. Sandini, Imitation learning of non-linear point-to-point robot motions using dirichlet processes, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2012, pp. 2029–2034.

[10] A. Pistillo, S. Calinon, D. Caldwell, Bilateral physical interaction with a robot manipulator through a weighted combination of flow fields, in: IEEE/RSJ Int. Conf. on Intel. Robots and Systems, 2011, pp. 3047–3052.

[11] T. Luksch, M. Gienger, M. Muhlig, T. Yoshiike, Adaptive movement sequences and predictive decisions based on hierarchical dynamical systems, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2012, pp. 2082–2088.

[12] F. Stulp, O. Sigaud, Robot Skill Learning: From Reinforcement Learning to Evolution Strategies, Paladyn. Journal of Behavioral Robotics 4 (1) (2013) 49–61.

[13] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Handbook of Robotics, Springer Berlin Heidelberg, 2008, Ch. Robot Programming by Demonstration, pp. 1371–1394.

[14] G. McLachlan, D. Peel, Finite Mixture Models, Wiley, 2000.

[15] C. Rasmussen, C. Williams, Gaussian processes for machine learning, Springer, 2006.

[16] S. Vijayakumar, A. D'Souza, S. Schaal, Incremental online learning in high dimensions, Neural Computation 17 (12) (2005) 2602–2634.

[17] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222.

[18] A. Soltoggio, A. Lemme, J. J. Steil, Using movement primitives in interpreting and decomposing complex trajectories in learning-by-doing, in: Proc. of IEEE Int. Conf. on Robotics and Biomimetics (ROBIO), 2012, pp. 1427–1433.

[19] N. Lawrence, Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data, in: In NIPS, 2003.

[20] Z. Artstein, Stabilization with relaxed controls, Nonlinear Analysis 7 (1983) 1163–1173.

[21] Z. Jiang, Y. Lin, Y. Wang, Stabilization of nonlinear time-varying systems: a control lyapunov function approach, Journal of Systems Science and Complexity 22 (2009) 683–696.

[22] E. D. Sontag, Mathematical Control Theory: Deterministic Finite Dimensional Systems, 2nd Edition, Springer, New York, 1998.

[23] P. Kokotovic, M. Arcak, Constructive nonlinear control: a historical perspective, Automatica 37 (2001) 637–662.

[24] A. E. Bryson, Y. Ho, Applied Optimal Control: Optimization, Estimation, & Control, Taylor & Francis; Revised edition, 1975.

[25] S. M. Khansari-Zadeh, A Dynamical System-based Approach to Modeling Stable Robot Control Policies via Imitation Learning, Phd thesis, cole Polytechnique Fdrale de Lausanne (November 2012).

[26] S. M. LaValle, Planning Algorithms, Cambridge University Press, 2006.

[27] S. Calinon, F. Guenter, A. Billard, On Learning, Representing and Generalizing a Task in a Humanoid Robot, IEEE transactions on systems, man and cybernetics 37 (2) (2007) 286–298.

[28] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. Journal of Robotics Research 5 (1986) 90–98.

[29] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, A. Billard, Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression, IEEE Robotics and Automation Magazine 17:2 (2010) 44–54.

[30] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives, IEEE Trans. on Robotics 26 (5) (2010) 800–815.

[31] L. Righetti, J. Buchli, A. J. Ijspeert, Dynamic hebbian learning in adaptive frequency oscillators, Physica D 216 (2006) 269–281.

[32] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, M. Kawato, Learning from demonstration and adaptation of biped locomotion, Robot. Auton. Syst. 47 (2004) 79–91.

[33] S. M. Khansari-Zadeh, K. Kronander, A. Billard, Learning to play minigolf: A dynamical system-based approach, Advanced Robotics 26 (17) (2012) 1967–1993.

[34] S. Calinon, E. Sauser, A. Billard, D. Caldwell, Evaluation of a probabilistic approach to learn and reproduce gestures by imitation, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2010, pp. 2671–2676.

[35] S. M. Khansari-Zadeh, A. Billard, A dynamical system approach to real-

time obstacle avoidance, Autonomous Robots 32 (2012) 433–454.
[36] D.-H. Park, H. Hoffmann, P. Pastor, S. Schaal, Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields, in: IEEE Int. Conf. on Humanoid Robotics, 2008, pp. 91–98.
[37] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, O. Sigaud, Learning Compact Parameterized Skills with a Single Regression, in: Proc. IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids), 2013, pp. 1–7.
[38] S. Schaal, A. J. Ijspeert, A. Billard, Computational Approaches to Motor Learning by Imitation, Philosophical Transactions: Biological Sciences (The Royal Society) 358 (1431) (2003) 537–547.
[39] D. Wolpert, M. Kawato, Multiple paired forward and inverse models for motor control, Neural Networks 11 (1998) 1317–1329.
[40] D. Bullock, S. Grossberg, The VITE Model: A Neural Command Circuit for Generating Arm and Articulator Trajectories, Dynamic Patterns in Complex Systems (1988) 305–326.
[41] P. Gaudiano, S. Grossberg, Adaptive vector integration to endpoint: Self-organizing neural circuits for control of plannedmovement trajectories, Human Movement Science 11 (1992) 141–155.
[42] M. Raibert, Legged robots that balance, Cambridge MA: MIT Press, 1986.
[43] A. J. Ijspeert, J. Hallam, D. Willshaw, Evolution of a central pattern generator for the swimming and trotting gaits of the salamander, in: Int. Conf. on Computational Intelligence & Neurosciences (ICCIN), 1998, pp. 24–38.
[44] M. Lukosevicius, H. Jaeger, Reservoir computing approaches to recurrent neural network training, Computer Science Review 3 (3) (2009) 127–149.
[45] R. F. Reinhart, J. J. Steil, Neural learning and dynamical selection of redundant solutions for inverse kinematic control, in: IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids), 2011, pp. 564–569.
[46] B. Pearlmutter, Learning state space trajectories in recurrent neural networks, Neural Computation 1 (1989) 263–269.
[47] A. F. Atiya, A. G. Parlos, S. Member, S. Member, New Results on Recurrent Network Training: Unifying the Algorithms and Accelerating Convergence, IEEE Trans. Neural Networks 11 (2000) 697–709.
[48] S. Schaal, Dynamic movement primitives - A framework for motor control in humans and humanoid robots, in: Int. Symposium on Adaptive Motion of Animals and Machines, 2003.
[49] B. Nemec, M. Tamosiunaite, F. Worgotter, A. Ude, Task adaptation through exploration and action sequencing, in: Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids), 2009, pp. 610–616.
[50] S. M. Khansari-Zadeh, A. Billard, BM: An iterative algorithm to learn stable non-linear dynamical systems with Gaussian mixture models, in: Int, Conf. on Robotics and Automation (ICRA), 2010, pp. 2381–2388.
[51] S. Calinon, A. Pistillo, D. G. Caldwell, Encoding the time and space constraints of a task in explicit-duration hidden Markov model, in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2011, pp. 3413–3418.
[52] K. Kronander, S. M. Khansari Zadeh, A. Billard, Learning to control planar hitting motions in a minigolf-like task, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2011, pp. 710–717.
[53] A. Shukla, A. Billard, Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies, Robotics and Autonomous Systems 60 (3) (2012) 424–440.
[54] K. Kronander, A. Billard, Learning compliant manipulation through kinesthetic and tactile human-robot interaction, IEEE Transactions on Haptics.
[55] S. Kim, Rapid and reactive robot control framework for catching objects in flight, Phd thesis, cole Polytechnique Fdrale de Lausanne (2014).
[56] A. Shukla, A. Billard, Augmented-SVM: Automatic space partitioning for combining multiple non-linear dynamics, in: Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1025–1033.
[57] J. Bonnans, J. Gilbert, Numerical Optimization, 2nd Edition, Springer, 2006.
[58] H. Khalil, Nonlinear systems, Prentice Hall Upper Saddle River, 1996.
[59] P. Kulchenko, E. Todorov, First-exit model predictive control of fast discontinuous dynamics: Application to ball bouncing, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2011, pp. 2144–2151.
[60] L. Yang, S. LaValle, The sampling-based neighborhood graph: an approach to computing and executing feedback motion strategies, IEEE Transactions on Robotics and Automation 20 (3) (2004) 419–432.

Seyed Mohammad Khansari-Zadeh is a post-doctoral researcher at LASA laboratory at EPFL. He received his Ph.D. (2012) in Robotics from EPFL, and his M.Sc. (2008) in Dynamics and Control from Sharif University of Technology. His research interests include imitation learning for robot control, dynamical systems, nonlinear control, and humanoids.

Aude Billard is Associate Professor and head of the Learning Algorithms and Systems Laboratory at the School of Engineering at EPFL. She received her B.Sc. (1994) and M.Sc. (1995) in Physics from EPFL and her Ph.D. in Artificial Intelligence (1998) from the University of Edinburgh.