Learning Externally Modulated Dynamical Systems

Nicolas Sommer, Klas Kronander and Aude Billard

***Abstract*— Dynamical Systems (DS) are often used to represent motion, with the advantage of being easy to learn from demonstrations. We present a method to modulate DS depending on an external signal, extending our previous work on Locally Modulated DS (LMDS [1]). We present two applications of our system, which would not have been possible to achieve without taking external sensing into account in the DS motion formulation. The first application is a task of localization and grasping of objects, using our previous work on compliant tactile exploration. We successfully localize and grasp objects whose position is unknown, using touch in a simulated environment. In the second application, we teach a robot how to react to collisions in order to navigate between obstacles while reaching.**

I. Introduction

With robots moving into human-centered environments, the use of sensory information becomes more and more important to interact with everyday objects. In order to generate robot motion, the traditional methods based on planning and execution are not well suited to uncertain and quickly changing environments. For instance, grasping traditionally relies on several distinct steps: computing a grasp configuration, planning a collision-free robot trajectory and executing that grasp [2], [3]. If the object moves, or the pose is uncertain, the whole process may have to be started over. Also, because planning methods can yield very different results with small configuration differences, the new planned trajectory might be completely different.

Dynamical Systems (DS) offer an efficient way to encode reaching [4] and grasping motions [5], which do not require to re-plan when the configuration changes. This allows to continuously and instantaneously update the trajectory. Furthermore, DS can be learned from demonstrations, instead of programming the robot explicitly. Instead of defining robot tasks as timed trajectories, or as dynamical systems that are indirectly driven forward by time, it is possible to define tasks as time-invariant dynamical systems. The latter have been shown to have numerous advantages for tasks that involve temporal and spatial perturbations [6].

In order to successfully model the robot motion, the possibility to incrementally perform the demonstrations allows the teacher to refine her demonstrations depending on the robot's current performance. In our previous work [1], a way to locally reshape an existing, stable nonlinear autonomous DS, while preserving important stability properties of the original system, was offered. This approach also included a method to enable incremental learning algorithm based on Gaussian Processes, for learning to reshape dynamical systems using this representation.

When executing a motion in a real environment, there is also a need to react to external sensory events, besides simply re-planning after a perturbation. For instance, when reaching for something and detecting contact with the robot arm, the trajectory of the robot may need to be adapted online, by modulating the arm dynamics depending on the sensed contact. One way to introduce a dependency from an external signal is through coupling across DS. However, we target here dependency on an external signal whose dynamics may not be known and hence cannot be done through coupling with another DS. Approaches to DS control with external sensing is used primarily for free-space motion and to update the state of the robot and the state of the attractor. Only a few attempts used an external sensing, force, as an input to the system [7], [8]. However, this was used to generate the desired trajectory and was then combined into a traditional impedance controller. Moreover, the sensing modulation was global. Here, we generalize this approach to enabling modulation from different types of sensing, not just force, and to allow the modulation to act locally, so as to provide modulation only in relevant parts of the task. Another approach consists in directly including external sensing to the inputs of the regression when learning a DS from demonstrations. By learning a mapping between end-effector position, tactile sensing, and velocity from demonstrations [9], we were able to generate behaviors based on tactile sensing, including grasping. However, because the resulting system is not autonomous and there are no constraints on the DS formulation, it is hard to ensure stability in this case. We address this by proposing a novel DS representation, called Externally Modulated Dynamical Systems (EMDS). We extend previous work (LMDS) to integrate external input in the DS and use it to reshape its dynamics. We also propose a method to learn how the dynamics are modulated depending on the external signal. Although introducing a dependency on an external signal, we can still guarantee preservation of the stability properties of the original dynamics.

In summary, the main contributions of this paper are:

1) Introduction of the EMDS framework, allowing the modulation of DS based on external signals while conserving important stability properties.
2) An interactive learning method for capturing *how* the DS should be modulated by the external signal.
3) The application of EMDS to several challenging tasks, including blind reach-and-grasp, using only tactile input for object state estimation, and navigating through obstacles using contact information only.

The remainder of this paper is organized as follows: In section II, we present related work. In section III, we introduce the EMDS formalism and a possible design of the modulation function. We also illustrate the possibilities offered by this formulation in several 2D examples. In section IV, we detail a complete framework used to autonomously localize and grasp objects, in which the EMDS plays a key role, with experiments and results. Finally, section V is dedicated to experimental validation on a different but equally important manipulation skill – navigation through unknown obstacles.

II. Related work

In robotics, Dynamical Systems (DS) have proven to be an interesting approach to motion generation, as an alternative to classical methods relying on separate planning and execution. They offer a simple way to integrate both steps into one formulation [10], [11].

Dynamical Movement Primitives (DMPs) have recently gained popularity [12], [13]. They are a set of differential equations that can compactly represent a large variety of robotic tasks. Their

The authors are with the Algorithms and Systems Laboratory (LASA), Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland.

mechanism also make it easy to incorporate in Reinforcement Learning, and learning without risking unstable behavior. They however rely on a phase variable acting as an implicit clock, forcing the system to converge to a linear system as the phase converges.

Autonomous DS formulations [6] allow to represent motions in a time-independent manner, in contrast with time-varying representations. Because stability is a major concern when dealing with DS, this has been addressed in our previous work [14] for a specific parametric form of DS, Gaussian Mixture Regression (GMR). To increase the flexibility of learned motions and to allow non-parametric learning, we later introduced LMDS [1]. This formulation does not base the stability analysis on a known Lyapunov function, therefore incremental demonstrations do not need to comply with an energy function. In this paper, we build upon this formulation to create DS with equivalent stability properties while depending on external signals.

Recently, [15] introduced a framework to react to sensory input while performing reaching-type motions with DMPs. In this work, the DMP's trajectory is adapted in order to match previously learned sensory signal, i.e. force information. This is done through a predefined mapping between sensing and end-effector accelerations, using the task Jacobian of the sensor. This approach is directly applicable to situations in which such mappings between sensory signals and control signal can be manually defined. This includes sensors with low-dimensional inputs such as force-torque sensors, but cannot be extended well to a tactile skin on multiple fingers for instance. For tactile data, it is generally not possible to define generic mappings from sensor signature to appropriate control response. More recently, the authors generalized their work in the Associative Skill Memory framework [16], which switches between learned motor primitives based on sensory signature (using hard switches). This is based on the assumption that task representations should be stereotypical with as little variation as possible in order for the associated sensory recordings to have little variance. In [17], the robot adapts its learnt behaviour according to a predefined low-level controller that depends on sensory input. In this paper, we suggest to learn the mapping from external signal to modulation of the dynamics, which are provided by a time-invariant DS.

In our previous work [18], we developed a compliant controller to adapt the robot's fingers to unknown shapes, with an application to grasping. Using tactile sensors to detect contacts, this controller maximizes the surface in contact, using the null-space of existing contacts to keep the contact forces low. In addition to classical tactile-based grasping controllers, this controller can create additional contacts on the same kinematic chain. For instance, while a finger is in contact at the fingertip, other contacts are made on the first phalanxes by sliding the fingertip on the surface, without loosing contact. This controller is used in the experiment presented in section IV in order to both increase the number of contacts points, thus speeding up the localization process, and to provide a compliant enclosing mechanism when the object is finally grasped.

III. Approach

A. Locally Modulated Dynamical Systems

In [1], the Locally Modulated Dynamical Systems (LMDS) formulation was introduced. Since this paper extends that work, we first provide a brief overview of the LMDS formulation in this section. LMDS allows to apply arbitrary local learning algorithms to reshape motion dynamics without loss of stability.

Let $x \in \mathbb{R}^N$ represent a N-dimensional kinematic variable, e.g. a Cartesian position or a joint angle vector. Let a continuous function $f\colon \mathbb{R}^N \to \mathbb{R}^N$ represent the original dynamics (OD): $\dot{x} = f(x)$. We define the *Locally Modulated Dynamical Systems* by multiplying the previous equation by a matrix-valued continuous function $M(x) \in \mathbb{R}^{N\times N}$, yielding:

$$\dot{x} = M(x)f(x) \tag{1}$$

Using modulation functions that depend only on the state variable $x \in \mathbb{R}^N$ allows us to prove a number of interesting properties of the reshaped dynamics, including boundedness preservation of stability properties [1]. Importantly, with an appropriate parameterization of the modulation function M, LMDS can be used with non-parametric learning algorithms without constraints.

B. Externally Modulated Dynamical Systems

In the LMDS formulation, the input to the system is the state of the robot, x. In many tasks, it is necessary to be able to react to sensory input in a task-specific manner. The goal of the Externally Modulated Dynamical Systems (EMDS) is to provide a DS formulation that allows to learn reactions to sensory events such as contact detected with tactile sensing arrays or force-torque sensors.

Let $s \in \mathbb{R}^M$ be a M-dimensional external signal, independent of the state of the DS. In EMDS, the dynamics are reshaped by a modulation field $M(x, s)$. The form of the dynamics follows the same reshaping structure as LMDS:

$$\dot{x} = M(x, s)f(x) \tag{2}$$

where $M(x, s) \in \mathbb{R}^{N\times N}$ is a continuous matrix valued function that modulates the original dynamics $f(x)$. The difference to LMDS in formulation is hence that we allow the modulation to be a function not only on the DS state but also on our external signal.

As the resulting DS is not autonomous, we cannot expect the same stability properties as in the case of the autonomous LMDS formulation. However, by constructing the modulation matrix appropriately, we can achieve guaranteed boundedness and convergence of the dynamics by ensuring that M is full rank and locally active. Stability properties can easily be derived by adapting the proofs in [1] to include the external signal:

- The reshaped dynamics has the same equilibrium point as the OD.
- The reshaped dynamics is bounded.
- Assuming that the equilibrium point is centered at the origin and that the OD is stable, the reshaped system is stable at the origin.
- The reshaped system is locally asymptotically stable at the origin.
- The reshaped dynamics has the same equilibrium point as the OD.

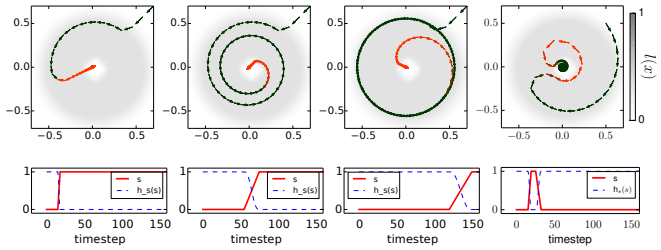
C. Design of the modulation function

The modulation field $M(x, s)$, as introduced in the previous section, has few design constraints. In this section, we introduce one possible way to design and parametrize this function.

1) Modulating rotating and speed-scaling dynamics: In [1], the modulation function was proposed to be defined as a composition of a speed scaling and a rotation matrix. It is always possible to represent this modulation function compactly as a parameter vector $\theta \in \mathbb{R}^L$, where $L \geq D$ will depend on the chosen parameterization and the dimension of the state $x \in \mathbb{R}^D$. Complex reshaping of the OD can then be learned by using non-linear regression to learn a function mapping from the state to this parameter vector.

In EMDS, we let the external signal s modulate the rotation angle and the speed scaling before reconstructing M and applying the modulation to the OD:

$$\theta(x, s) = h_s(s)[\phi(x)\mu_R, \kappa(x)] \tag{3}$$



(a) Early s chg. (b) Late s chg. (c) Limit cycle (d) Different OD

Fig. 1: **Top:** Examples of the DS modulated by an external signal. **Bottom:** Corresponding profiles of the external signal s and the function $h_s(s)$ which inhibits the rotational modulation. The DS is the same in (b)-(a) but the external signal's profile is different in each plot. (c) Example of a modulation of the DS, using $\phi_c = 90°$. The system does not converge while s is 0 (i.e. the external activation function $h_s(s)$ is 1). If that is the case, the system stays in a limit cycle. (d) An example with different OD and a modulation that also applies a speed-scaling.

with μ_R the normed rotation vector defining a rotation axis. The mappings $\phi(x) : \mathbb{R}^N \rightarrow [-\pi, \pi]$ and $\kappa(x) : \mathbb{R}^N \rightarrow \mathbb{R}^+$ are continuous functions from a robot state to respectively a rotation angle and a speed-scaling. As in standard LMDS [1], the state dependent maps $\phi(x)$ and $\kappa(x)$ should be locally active. The parameters are also influenced by the continuous external activation function $h_s : \mathbb{R}^M \rightarrow [0, 1]$, which depends on the external signal s. The modulation function $M(x, s)$ is defined as:

$$M(x, s) = \kappa(x, s)R(x, s) \tag{4}$$

with $R(x, s)$ the associated rotation matrix, hence full rank by construction. So does $M(x, s)$ and therefore all the stability properties are guaranteed for any x and s.

D. Illustrative examples

Here, we give a few 2d illustrative examples of how the external input in EMDS can influence the dynamics of the original DS, using the modulation matrix design presented in the previous section. Consider the following linear original dynamics:

$$\dot{x} = -Ax = -\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} x \tag{5}$$

Let the function $l : \mathbb{R}^2 \rightarrow [0, 1]$ describe the influence of the modulation and impose the locally active property around 0. The value of $l(x)$ is visible in Grey-scale for the examples in fig. 1. The external signal s influences the local modulation according to a smooth activation function $h_s : \mathbb{R} \rightarrow [0, 1]$ from 1 to 0, with $h_s(0) = 1$ and $h_s(1) = 0$.

Introducing the local activation function $\phi(x) = l(x)\phi_c$, with $\phi_c \in [-\pi, \pi]$ a constant angle, the rotation angle $\Phi(x, s)$ is given by $\Phi(x, s) = h_s(s)l(x)\phi_c$. This results in a spiraling behavior where and when the DS is modulated. When the external signal is active, the rotation is inhibited and thus the system converges much faster with the OD – a straight line.

The resulting dynamics are given in figs. 1(a) and 1(b) using $\phi_c = 81°$ and different arbitrary profiles of external signal s. The evolution in time of the external signal s and consequently of the activation function $h_s(s)$ is drawn on the plots below the evolution of the DS in 2d, and the value of s is also represented on the top figures by the color of the arrows. When the signal s becomes high, the activation function $h_s(s)$ goes to 0 and the system switches from spiraling to the original linear dynamics, converging rapidly. The resulting behavior can be used for instance to switch between searching and reaching motions on a robot. In fig. 1(a), s is activated early and so do the dynamics, changing from spiraling to reaching directly. In fig. 1(b), s is activated late and at a slower rate, hence the system follows the modulated dynamics, spiraling, for a long time until gradually changing. We also provide an example where the system is not globally asymptotically stable in fig. 1(c), by setting the maximum modulation angle ϕ_c to 90°. When the external signal is 0, the DS goes into a limit cycle. Boundedness is however enforced thanks to the locally active property. In fig. 1(d), we illustrate another example behavior of our modulated system using different original dynamics (with $A = \begin{bmatrix} 0.05 & 0.2 \\ -0.2 & 0.05 \end{bmatrix}$ in eq. (5)), a maximum modulation angle $\phi_c = 160°$ and a signal s varying between 0 and 1. The modulation also applies a speed-scaling of factor 3, visible on the top figure from the length of the arrows changing with s. Depending on s, the direction of the rotation is changed.

a) Comments on local modulation: The local property ensures boundedness and local asymptotic stability for any chosen modulation matrix. It may thus be useful to keep locality even when not required by the desired dynamics, but only for the provided stability purposes. In a searching task such as illustrated in the above examples, it also makes sense to only activate the searching behavior in a subregion of the state-space corresponding to the searching region, hence the local activation around the attractor at 0.

E. Learning EMDS

Using the design of the modulation function presented above, it is possible to retrieve a normal LMDS by removing the dependency on the external signal, i.e. by replacing $h_s(s)$ with 1 (i.e. never inhibiting the local modulation). Conversely, an EMDS can be created by associating an existing LMDS with the function $h_s(s)$.

Therefore, an EMDS can be based on an LMDS learned the same way as in the original LMDS formulation, using GP-MDS based on Gaussian Processes Regression (GPR), or any arbitrary local learning algorithm. The external signal activation function $h_s(s)$ can then be provided or learned separately to form the EMDS. To sum up, one way to learn a complete EMDS from scratch with training data can be the following procedure:

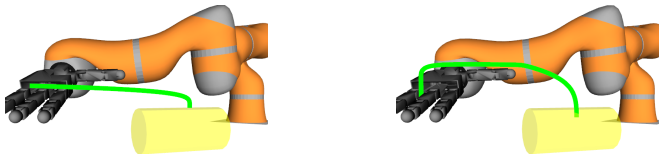
1) Learn a DS, the original dynamics, from demonstration data, e.g. with SEDS [14].
2) Learn new dynamics from other demonstration data to represent different dynamics, expressed as a modulation of the OD, using a local learning algorithm or GP-MDS as presented in [1].
3) Learn the function $h_s(s)$.

See the following sections IV-A and V-A.2 for examples of learning $h_s(s)$ as a non-linear regression problem using Gaussian Process and human demonstrations to train the model on.

IV. Experiment 1: Autonomous Localization and Grasping

As an application of EMDS, we consider a highly challenging autonomous search-and-grasp task. We rely entirely on tactile sensing to localize the object to be grasped. In such a task, there are two subtasks involved: 1) to estimate the pose of the object and 2), when certain enough of the object's pose, to attempt to grasp the object. The searching and grasping behaviors are first modeled by LMDS (search as the original dynamics, and grasp as the reshaped dynamics, see fig. 2). Then, to incorporate our confidence in the object pose's estimate in the dynamics, we learn an activation function $h_s(s)$ from demonstrations and use the EMDS as described in section III.

During the exploration motion of the arm, tactile data (position and normal) from contacts with the robot is fed to a particle



(a) Original dynamics (b) Modulated dynamics

Fig. 2: **Exp 1:** In green, reaching trajectories for the original and modulated dynamics. The modulated dynamics force the trajectory to approach the object from above in order to avoid collisions with the fingers.

filter (PF) responsible for localizing the object. The state of the estimation is fed back to the EMDS: the variance of the object pose's distribution is used to modulate between searching and grasping behaviors, see fig. 3. The hand and finger's behavior is controlled by a coupling mechanism [5] between the EMDS generating the arm motion and a coupled DS generating the finger motion, which is given as a subtask to our contact controller from [18]. This controller provides a compliant mechanism to maximize contact points, thus speeding up the localization process.

We use the design presented in section III-C.1, based on modulating rotation dynamics, to generate the arm motion. The attractor of the EMDS is translated to the latest expected value of the robot's pose, i.e. the weighted average over each particle from the particle filter (PF). The external signal is chosen as the norm of the covariance matrix of the object's position estimate.

Both the original dynamics (OD) and the modulation of the LMDS are provided as modeled reaching and grasping behaviors[1]. A searching behavior is implicitly provided by the evolution of the PF's best estimate. When the hand goes to a location and there is no sensed contact, the particles become depleted in that area. The robot then goes to the new best estimate at a different location and this process results in a searching behavior. An illustration of this behavior can be seen in fig. 6.

In the modulated trajectory corresponding to the grasping behavior, the hand approaches the current attractor from above in order to implicitly avoid collisions with the fingers and to properly enclose the object. Typical trajectories of the original and the modulated dynamics can be seen on fig. 2. The output of the EMDS is the desired velocity of the end-effector, connected to a passive DS controller [19], designed to perform closed-loop control of DS while ensuring passivity, and ideally suited for uncertain manipulation tasks such as this.

A. Learning the external activation function

The function $h_s(s)$ mapping the external signal to the activation of the modulation can either be programmed or learned, with the constraint that its values lie between 0 and 1. We chose to learn this function since the values of the covariance matrix are not necessarily easy to link to the task. On the contrary, our graphical visualization (see fig. 6 for instance) of the object's pose estimated distribution is easy to interpret. It provides a way for the user to perceive the current particle's filter uncertainty value s through the visualization of the particles. In order to learn the function $h_s(s)$, we go through a short learning phase during which a teacher manually selects the desired behavior while the task is being executed. During this phase, the teacher chooses how much to inhibit or not the local modulation of the original LMDS system using a graphical user interface (GUI) with a slider control. The teacher chooses continuous values between 0 (reaching approach) and 1 (grasping approach).

The recorded data are used to train a Gaussian Process Regression (GPR) model, using the squared exponential covariance function. The kernel's hyper-parameters are determined manually by using prior knowledge from the training data. Such a learned function $h_s(s)$ and the training data can be seen on fig. 4.

The value of h_s starts to increase from 0 to 1 when s is below 0.2. During runtime, we use the stored GP model to predict the value of h_s given an input s.

[1]In our case, the OD – reaching – is a simple linear DS. The grasping behavior is coded as a modulation of the OD to generate a grasping behavior, approaching the object from above. While the motion for search and grasping can be learned using existing methods (as described in section III-E), the biggest challenge of a blind search-and-grasp task is how to switch between these two behaviors. This is related to the exploration/exploitation trade-off in reinforcement learning. Here, we use a human to support the acquisition of this skill.

B. Experimental setup

The simulation environment is Gazebo, with a Kuka LWR robotic arm with 7 degrees of freedom (DOF) and 16-DOFs AllegroHand, see fig. 2. All 23 DOFs of the system are torque-controlled.

The PF for estimating the object's pose requires the evaluation of the likelihood of a measurement for a potential object pose. This is achieved by generating a virtual measurement for the virtual object pose of each particle. This virtual measurement is generated by a second instance of Gazebo, running a copy of the simulated world and keeping the robot's configuration synchronized. For each particle and corresponding object position, the object is moved in the second world and the state of the contacts is updated to compare it with the real simulated world.

We were able to run these measurements at a rate of about 1000 particles per second on one thread with a Core i7 cpu. This step being the bottleneck for the PF's update rate, our 300-particles filter could run roughly at 3Hz.

The objects used in this experiments are presented on fig. 5(b), we begin the experiment with a simple cylinder, then a more complex artificial object, non convex, in the shape of a cross composed of cylinders and spheres, and a drill.

For each trial, the object's pose ξ is randomly generated in the simulation environment from a uniform distribution in a plane: $\xi \in [-x_l, x_l] \times [-y_l, y_l] \times [-\theta_l, \theta_l]$, ($x_l = 0.1$, $y_l = 0.1$, $\theta_l = \frac{\pi}{2}$).

C. Results

We compare the EMDS with using LMDS only, i.e. without taking into account the external signal. This leads to the dynamics always following the modulated system, i.e. the grasping approach. For each tested condition, EMDS or LMDS, and object, cylinder, drill or cross, the trials are carried out 50 times in simulation. Each trial lasts 30 seconds. This represents 150 minutes of simulated experiments. The results are reported on fig. 5(a) with boxplots.

We measure both the time to estimate the object's position and to reach to that position, with a threshold of 1.5cm. For each of the explored objects, the EMDS strategy estimates and reaches the real object's position significantly faster than with LMDS. The estimation takes $14.7 \pm 6.2s$ with LMDS, while only $9.6 \pm 5.6s$ with EMDS. It takes a little more to reach the object, with $15.7s \pm 5.0$ for LMDS and $11.0s \pm 4.9$ for LMDS. Because with LMDS, the robot tends to perform a grasping approach even though the object's pose is not known with certainty, it is not surprising that this method takes more time.

One example of a trajectory for localizing and grasping the cylinder can be seen on fig. 6. The particles are represented as red or black dots, the color representing their current respective weight. The current object's real and estimated positions are represented

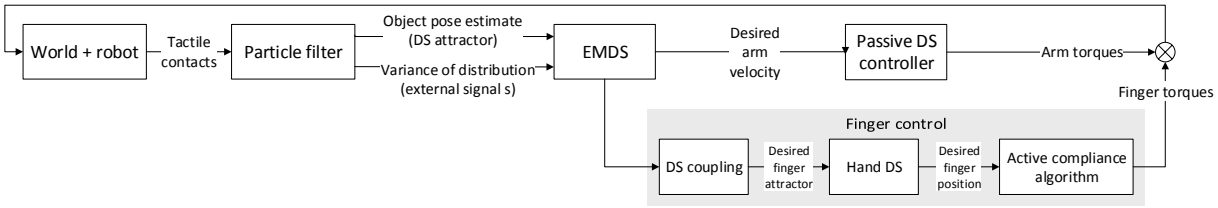


Fig. 3: The framework for autonomous localization and grasping with EMDS.

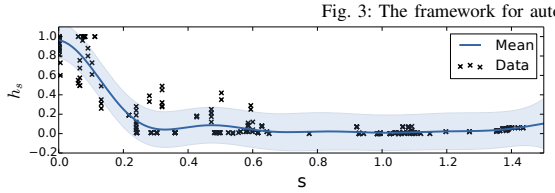


Fig. 4: Learning $h_s(s)$ from demonstration data in **Exp 1**. The blue envelope around the mean represents the variance of the GP function.

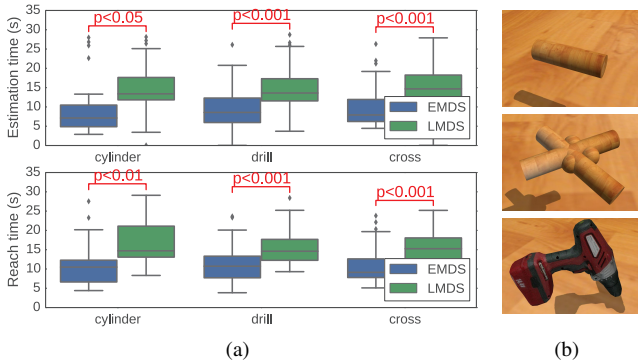


Fig. 5: **Exp 1. a)** Results of the experiments: time to estimate and time to reach the object's position using LMDS only or EMDS, for the three objects. **b)** The three objects used in **Exp 1.**

by RGB frames. The estimated trajectory given the current external signal can also be seen as a thick black line. The corresponding evolution of the external signal s, activation function $h_s(s)$ and the altitude of the hand during the exploration are given in fig. 7. On frames (**a**)-(**c**), the hand approaches the current estimated pose of the object directly, until it touches the object, the number of contacts increases, and the object's current estimated position is updated. On frame (**d**), the particles have all gathered at the real object's position, hence the variance of the object's estimated position distribution decreases and h_s increases. Therefore, the DS automatically adapts to a grasping motion, the altitude increases and the hand approaches the object from above until grasping it on frames (**g**)-(**h**).

In this experiment, we showed that by training an EMDS, we could teach the desired behavior of the robot depending on an external signal, here the confidence over the object's pose from a PF, using only tactile data.

V. Experiment 2: Reaching while avoiding obstacles

In this section, we present another application of our algorithm in which a task of reaching while avoiding obstacles is encoded. The task consists in going from point A to point B for the robot end-effector with the desired dynamics, while there are obstacles with unknown position on the path.

We present two experiments: in the first experiment, the avoidance behavior is achieved by going over the obstacles. This is demonstrated both in simulation and on the real robot. In the second experiment, the robot's avoiding behavior depends on characteristics of the collision: it takes a different trajectory depending on the detected angle of the collision with the object. The second experiment is performed on the robot.

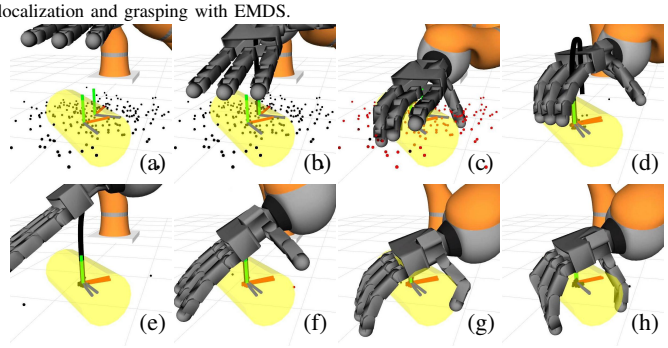


Fig. 6: **Exp 1:** Images of the exploration and grasping procedure with the cylinder. On frame (**d**), the hand automatically switches to a grasping motion, and moves up to approach the object from above, until grasping on frame (**h**). The references frames made out of three RGB segments represent the real object's position and the estimated one. The estimated object's pose can be seen moving towards the real one as the PF converges using tactile information

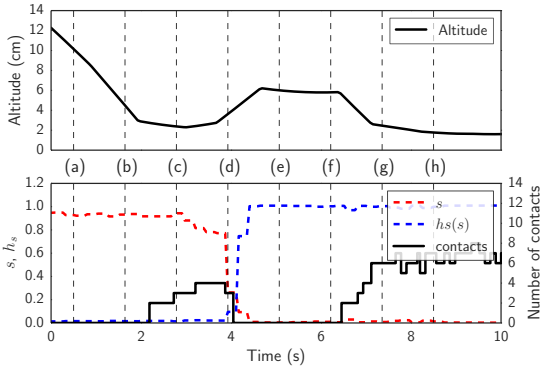


Fig. 7: **Exp 1:** Evolution of the external signal s, activation function $h_s(s)$ and the altitude of the hand during one exploration trial. The labels (a)-(h) correspond to the steps of exploration in fig. 6.

A. Experiment 2a: Avoiding obstacles

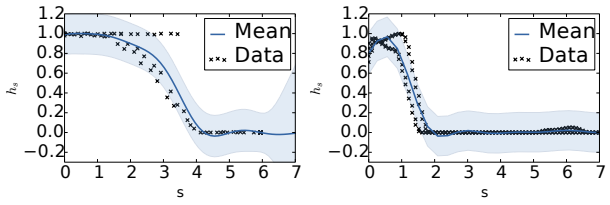
The original dynamics used in **Exp 2a** are simple linear dynamics – reaching in a straight line – while the modulated dynamics correspond to a maneuver to avoid obstacles. Instead of trying to reach through obstacles, the modified dynamics encode trajectories for which the end-effector moves back and goes over them, thus doing a detour while still reaching for the target at point B (see fig. 8). This strategy allows to reach directly to the target while only taking a detour when necessary. The start and end points are $60cm$ away from each other on the y axis. Obstacles are placed on the path, see figs. 10 and 11.

Similarly as in **Exp 1**, the function $h_s(s)$ mapping the external signal to the activation of the modulation is learned from demonstrations. Because the robot cannot stay in contact when there is a collision, and the system does not have a memory of the last contact, we choose to encode as the external signal s the time since the last collision. This allows the system to learn how to avoid obstacles given the memory of the last time a collision occurred.

1) Learning the activation function from GUI demonstrations: During the execution of the task **in simulation**, a teacher specifies continuous values for the modulation signal $h_s(s)$ between 0 (reaching directly) and 1 (reaching indirectly with the avoiding

(a) Original dynamics (b) Modulated dynamics

Fig. 8: **Exp 2a:** In green, reaching and avoiding trajectories corresponding to the original and modulated dynamics. The RGB frame on the right corresponds to the target. The modulated dynamics force the end-effector to go above obstacles and avoid collision with it.



(a) h_s simulation (b) h_s real robot

Fig. 9: **Exp 2a:** Learning $h_s(s)$ from demonstrations. The blue envelope around the mean represents the variance of the GP function. The external signal s corresponds to the time since the last collision.

maneuver). The learned function $h_s(s)$ and the training points can be seen on fig. 9(a).

The learned modulation function corresponds to choosing the avoiding dynamics ($h_s(s) = 1$) when a contact occurred less than 2 seconds ago, and slowly switching to the original linear dynamics until 4 seconds after a contact. Then, the system converges again in a straight line towards the target ($h_s(s) = 0$).

2) Learning the activation function from real robot demonstrations: **On the real robot**, we make an attempt at learning the function $h_s(s)$ from demonstrations, by back-driving the robot arm. In simulation, there is no practical way to let a human guide or teleoperate a robot easily while providing feedback, for instance about collisions. Yet, it is crucial for the user to be aware of the external signals on which depends the activation of the modulation.

On the real robot, we can however let a human perform demonstrations by back-driving the robot[2]. The objective is to define a mapping from the external signal s, chosen in this experiment as the time since last contact, to an activation value h_s. However, this value is only implicitly given by the user through the demonstrated trajectories. Hence, at each timestep, we need to find the value h_s which produces the demonstrated velocity $\dot{x}_s$ closest to the demonstrated velocity $\dot{x}$.

One way is to look for an inverse mapping of the EMDS. s is always expressed through the activation value h_s, as in eq. (3) for instance. The whole DS can then be expressed for a fixed x as a function: $\dot{x} = G_x(h_s)$. If the function G_x is injective, there exists an inverse function G_x^{-1} to retrieve h_s from $\dot{x}$ at a given x. We know with certainty that the function G_x is not injective for all x. For instance at $x = 0$, $f(x)$ is 0 since the OD is stable at the origin, hence G_x cannot be injective.

Instead of computing a closed-form inverse function, we estimate the value of h_s at each timestep by performing a line-search in the input space of $G_x(h_s)$, and search for the closest output:

$$h_s = \underset{h_{s\,i} \in \{0,\dots,1\}}{\operatorname{argmin}} \|G_x(h_{s\,i}) - \dot{x}\| \quad (6)$$

We perform the line-search with values from 0 to 1 by increments of 0.1, and interpolate between the two closest values.

Finally, we learn the function $h_s(s)$ in the same way as in the method demonstrated previously, using Gaussian Process Regression. The result can be seen on fig. 9(b).

In comparison with the function learned in simulation, the activation function goes back to zero only after 2 seconds, instead of 4 seconds. The slope of the decrease is also much more abrupt. This is probably due to the different method of teaching: while in the first case teaching is done through a graphical interface with a slider, in this case the user directly back-drives the robot arm.

3) Results, experiment 2a:

a) In simulation: The progression of the task can be seen on fig. 11. The corresponding evolution of the external signal s, predicted modulation $h_s(s)$, altitude of the hand, progression towards the target, and collision status are given on fig. 12. The predicted modulation $h_s(s)$ directly depends on s, following the learning process described above. The altitude of the hand depends mostly on the modulation or not of the dynamics: if the activation function is high, the hand follows the avoidance dynamics and altitude increases. The progression towards the target is the y coordinate normalized between 0 (start point) and 1 (target point). Objects from the first experiment (see fig. 5(b)), namely the drill and the cylinder, are used as obstacles.

The change between the two dynamics can be seen through the evolution of several variables on fig. 12. When a contact occurs, s is reset to 0, and $h_s(s)$ goes to 1. Simultaneously, the altitude starts increasing and the progression of the task starts regressing, as the avoidance behaviour is triggered. Then, as the time since the last contact increases and goes over 2 seconds, $h_s(s)$ goes back to 0 and the altitude decreases: the end-effector tries to reach for the object directly again.

b) On the real robot: For the experiment on a real platform, we use a KUKA LWR robot with 7 DOFs with a force-torque sensor mounted at the end of the arm. We add a probe-like end-effector after the sensor to be the contact point during collisions.

[2]In standard position controlled/time dependent systems, it is not straightforward to handle demonstrations while the robot is already moving. Thanks to our passive DS controller, users can safely interact with the robot during task execution. Our controller also guarantees safety for the user: the max velocity of the DS is set to $7cm/s$, and with the largest eigenvalue of the damping matrix for the passive DS controller set to $130N/m.s$, the maximum end-effector force is $9.1N$.

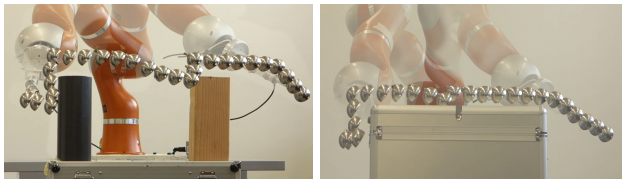


Fig. 10: **Exp 2a real robot:** Trajectory of the end-effector during the task with two objects or one large obstacle.

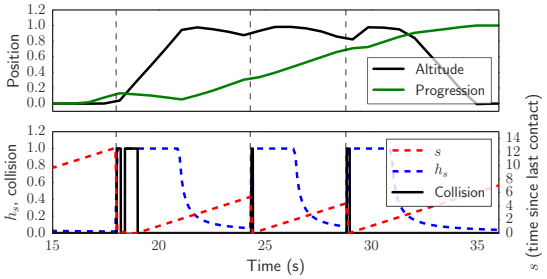


Fig. 12: **Exp 2a simulation. Top:** Evolution of the altitude of the hand (normalized) and the progression of the task (horizontal position between start and end points, normalized between 0 and 1). **Bottom:** Evolution of the external signal s (time since last collision), activation function $h_s(s)$, and the collision status. The vertical dotted lines correspond to collisions with obstacles, see fig. 11.

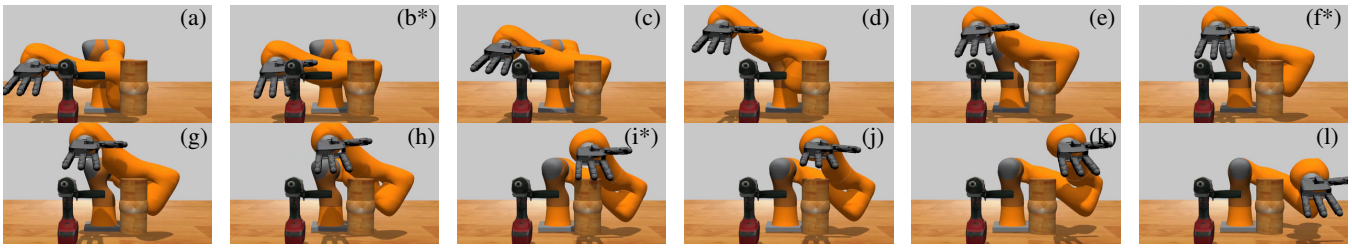


Fig. 11: **Exp 2a simulation:** Progression of the experiment in Gazebo. Collisions occur on frames **(b)**, **(f)** and **(i)**. Corresponding evolution of the variables on fig. 12.

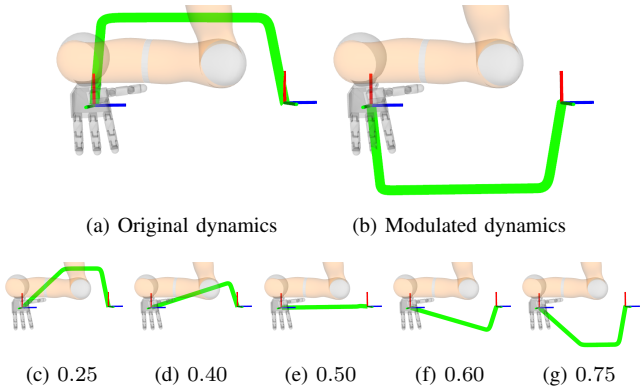


Fig. 13: **Exp 2b:** The RGB frames on the left and right correspond respectively to the starting point and the target. Seen from above, in green, trajectories of the DS. **Top:** The original and modulated dynamics. **Bottom:** The resulting dynamics with different levels of activation h_s. With $h_s = 0.50$, the dynamics reach in a straight line.

The force-torque sensor is used to detect collisions. We ran the same experiment on the real robot. We used objects from different sizes as obstacles. The images of the experiments can be seen on fig. 10. In the first one, the end-effector collides with the first obstacle, then with the second obstacle. In the second one, the end-effector collides first with the large box, then again on top of the large box.

Using external sensing only, we showed that the robot was able to avoid obstacles when contact is detected, following behavior taught during demonstrations.

B. Experiment 2b: Navigating between obstacles

In this experiment, we increase the complexity of the external modulation by taking into account two variables: the time since last contact, and the angle of the last contact. We aim at learning how to avoid obstacles depending on information from the collision, here the force direction during contact.

For this purpose, we encode the original dynamics and the modulated dynamics as two opposite velocity fields in a central region, where the experiment is taking place. Both dynamics converge to the target when going far enough from that region. The first one is directed perpendicular to the direction between initial and target frames, in a horizontal plane. The second one is directed in the opposite direction, see fig. 13. A whole range of dynamics is reachable by changing the activation of the modulation. For instance, by setting the activation to 0.5, the resulting trajectory is a straight line. The angle of the deviation can be adjusted by modifying the activation value between 0 and 1.

a) Learning the activation function: In order to learn the mapping $h_s(s)$, we perform demonstrations on the robot in a similar way as presented in the previous experiment. Because the input variable s is now two-dimensional (time since last contact and angle of last contact), more demonstrations must be given, spanning the whole input space. For this purpose, we perform 8 demonstrations with different collision angles.

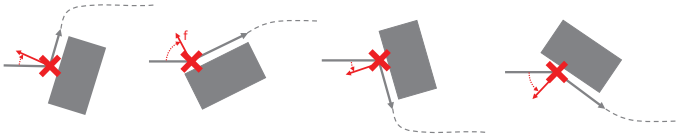


Fig. 14: **Exp 2b:** Schematic of the demonstrated trajectories. In red, the collision point and the force sensed during the collision. The end-effector follows the shape of the objects after contact, then continues again in a straight line after a few seconds.

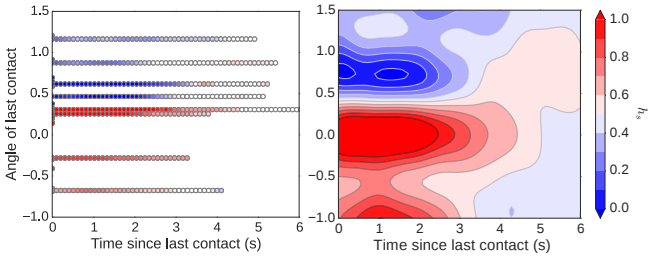


Fig. 15: **Exp 2b. Left:** Data from the demonstrations. Each horizontal line corresponds to one demonstration. **Right:** Learned function $h_s(s)$, using Gaussian Process Regression (GPR).

The behaviour taught to the robot is the following. When no collision occur, the robot moves in a straight line, i.e. $h_s(s) = 0.5$. After a collision, the end-effector adjusts its trajectory depending on the collision angle (see fig. 14). If the angle is small, the robot does a large detour, hence picks an extreme value of the activation function (0 or 1 depending on the direction of avoidance). Demonstration data are plotted on fig. 15(a). Each horizontal line of datapoints corresponds to a demonstration. We can see that the input space corresponding to the angle of collision is not perfectly spanned by the demonstrations, due to the demonstrated collision angles not being spread perfectly evenly. Looking at the horizontal axis, we can see that we recorded about 5 seconds of data after a collision per demonstration. The color of each datapoint indicates its corresponding h_s value, estimated using the method described in section V-A.2. We then learn a model with GPR, illustrated on fig. 15(b).

From these plots, we can extract a few observations: a) The collision angles are not centered on 0. The median value, for which the output $h_s(s)$ switches from values below 0.5 to values above, is at about $0.4rad$. It corresponds to a frontal collision. This is due to a mis-calibration of the force-torque sensor's orientation. Thanks to the learned mapping, this is not an issue. b) After a few seconds, all datapoints converge back to a value of $h_s(s) = 0.5$, i.e. a straight line[3]. This is the desired behaviour. c) The learned mapping, visible on fig. 15(b), corresponds to the desired behaviour: small angles (close to the median value of 0.4) yield extreme values of h_s, i.e. trajectories close to either one of the original or the modulated dynamics. Bigger angles yield less extreme values, leading to less modified trajectories.

b) Results: The task is executed both activating the modulation depending on the input s (fig. 16(a)), or fixing the value

[3]Output values are centered on 0 before learning, due to the GP values falling back to 0 far from demonstrated data.

(a) $h_s(s)$ learned (b) $hs(s) = 0.5$ (fixed)

Fig. 16: **Exp 2b:** Evolution of the obstacle avoidance task with the controller activated (a) or not (b). The end-effector reaches from left to right. In both cases, the desired velocity is tracked using a passive DS controller.

of the activation to 0.5, hence ignoring external signals for a comparison purpose (see fig. 16(b)). When the external signal is ignored, the end-effector moves in a straight line. Because of the passive DS controller, the robot is compliant. However, when ignoring the external signal, here the contact information, the robot does not avoid the obstacles. The friction in the joints prevents it to be really deviated from its trajectory, and the robot displaces the obstacles while colliding with them. When using information from the external signal, the robot adapts its trajectory after each collision and navigates between the obstacles. Depending on the collision angle, the robot adapts the avoidance trajectory. Therefore, it sometimes slides along the object (here with the second obstacle), or moves away from it (first and third obstacles).

VI. Discussion

Here, we provide a practical framework for achieving robust manipulation skills using external feedback. We have demonstrated that EMDS can easily be tailored to different tasks.

Desired behavior is achieved by modifying local modulations applied to an existing dynamical system. This allows to conserve important stability properties for any external signal, assuming that the modulation function is full-rank. Our proposed modulation function design ensures that the modulation matrix is always full-rank, and hence the system is stable for any external signal. As such, this work can be directly applied to an existing LMDS, provided a mapping between an external signal and the desired regulation of the modulation. For this purpose, we also suggested a method to capture how the dynamical system should be modulated by the external signal, based on learning the corresponding mapping by teaching the desired behavior during task execution.

We applied this algorithm to a task of simulated blind reach-and-grasp, using only tactile data for estimating the object's pose, which is in general extremely difficult to use in practice. In this task, the modulation between the reaching and grasping behaviors was encoded as a learned function of the variance of the pose estimate. The regulation of the modulation allowed to find and grasp the object faster than when always modulating the DS and following the grasping motion. To the best of our knowledge, this is the first time that blind search-and-grasp has been achieved, without using any vision.

We also applied this algorithm to tasks of reaching while avoiding obstacles. The system learns when to bypass obstacles depending on the last contact. We show that the task execution depends on learning a proper activation function, otherwise the behaviour is inadequate. The learned function depends on the time since the last contact, and thus depends implicitly on the size and shape of the obstacles seen during the task. The robot would perform less well with different obstacles as it would either collide again before bypassing (bigger obstacles), or make unnecessarily big detours (smaller obstacles). The teaching hence depends on the type of obstacles met during a specific type of task. We further studied this task by introducing the angle of collision into our activation mapping. With this two-dimensional external signal, our robot is able to navigate between obstacles and choose its trajectory by adjusting the level of activation of the DS's modulation, depending on the external signals.

In this work, the mapping from external signal to modulation is learned by teaching. We specify that the function can also be provided manually. An alternative to learning this function by a teacher would be to use reinforcement learning (RL) because the dimension of the problem is low and hence the problem fits particularly well the RL framework.

Acknowledgment

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Research (NCCR) Robotics, and the European Commission (Horizon 2020 Framework Programme, H2020-ICT-643950) through the Second-Hands project.

References

[1] K. Kronander, M. Khansari, and A. Billard, "Incremental Motion Learning with Locally Modulated Dynamical Systems," *Robot. Auton. Syst.*, 2015.

[2] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *ICRA*, 2000.

[3] M. A. Roa and R. Surez, "Grasp quality measures: review and performance," *Autonomous Robots*, 2014.

[4] S. Mohammad Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robot. Auton. Syst.*, 2014.

[5] A. Shukla and A. Billard, "Coupled dynamical system based armhand grasping model for learning fast adaptation strategies," *Robot. Auton. Syst.*, 2012.

[6] E. Gribovskaya, S. Khansari-Zadeh, and A. Billard, "Learning Non-linear Multivariate Dynamics of Motion in Robotic Manipulators," *Int. J. Rob. Res.*, 2011.

[7] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in *ICRA*, 2011.

[8] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task Parameterization Using Continuous Constraints Extracted From Human Demonstrations," *IEEE Transactions on Robotics*, 2015.

[9] N. Sommer, "Learning with tactile feedback on a humanoid robot." Master thesis, INSA de Strasbourg, 2012.

[10] A. Billard and G. M. Hayes, "DRAMA, a Connectionist Architecture for Control and Learning in Autonomous Robots," *Adapt. Behav.*, 1999.

[11] A. I. Selverston, "Are Central Pattern Generators Understandable?" *Behavioral and Brain Sciences*, 1980.

[12] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2003.

[13] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Computation*, 2013.

[14] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models," *IEEE Transactions on Robotics*, 2011.

[15] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *IROS*, 2011.

[16] P. Pastor, M. Kalakrishnan, F. Meier, F. Stulp, J. Buchli, E. Theodorou, and S. Schaal, "From dynamic movement primitives to associative skill memories," *Robot. Auton. Syst.*, 2013.

[17] O. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, 2010.

[18] N. Sommer and A. Billard, "Multi-contact haptic exploration and grasping with tactile sensors," *Robot. Auton. Syst.*, 2016.

[19] K. Kronander and A. Billard, "Passive Interaction Control With Dynamical Systems," *IEEE Robotics and Automation Letters*, 2016.