# Representing Movement Primitives as Implicit Dynamical Systems learned from Multiple Demonstrations

Robert Krug*
robert.krug@oru.se
*AASS Research Center, Örebro University, Sweden

Dimitar Dimitrov‡*
dimitar.dimitrov@inria.fr
‡INRIA Rhône-Alpes, 38331 St Ismier Cedex, France

***Abstract*—This work deals with the problem of parameter estimation of dynamical systems intended to model demonstrated motion profiles for a system of interest. The regression problem is formulated as a constrained nonlinear least squares problem. We present an approach that extends the concept of dynamical movement primitives to account for multiple demonstrations of a motion. We maintain an implicit dynamical system that resembles the demonstrated trajectories in a locally optimal way. This is achieved by solving a quadratic program (that encodes our notion of resemblance) at each sampling time step. Our method guarantees predictable state evolution even in regions of the state space not covered by the demonstrations.**

## I. Introduction

Parameter estimation[1] of dynamical systems is a fundamental problem in the context of processes where a mathematical model and corresponding experimental data are available. In robotic applications, like planning of reaching and grasping motions, the process of interest is the desired evolution of the state of a mechanical system ($e.g.$, a robotic gripper). To facilitate automatic planning schemes, data is often collected in form of demonstrations provided by a human expert. The goal is to generate "appropriate" motion patterns for the mechanical system (given a set of boundary conditions). In the framework of imitation learning, also referred to as Programming by Demonstration, the demonstrations are used to specify a desired transition from a given initial to a given final state in an intuitive way [2].

In the above context, mathematical models are selected based on their ability to generalize over multiple demonstrations while guaranteeing certain structural properties, and their potential to express coupling between the dynamics of different subsystems. Also, in order to facilitate the parameter estimation problem, simple models are often preferred.

On one hand, the use of Dynamical Systems (DS) for modeling of state transitions is beneficial compared to spline-based methods [3], [4], because a DS constitutes a policy over the state space and thus provides robustness to perturbations occurring during motion execution. On the other hand, choosing an appropriate policy ($i.e.$, dynamical system) might be problematic in light of the fact that usually the provided demonstrations are relatively sparse. Hence, it might happen that the behavior of the DS in "unexplored" parts of the state space is unexpected/undesirable. A classical approach for dealing with this problem is to enforce certain structural properties of the DS such as Global Asymptotic Stability (GAS), ensuring that the state is guaranteed to (at least) converge to the global equilibrium point. One shortcoming of such an approach is that it does not state any preference about the behavior of the system in relation to the demonstrations.

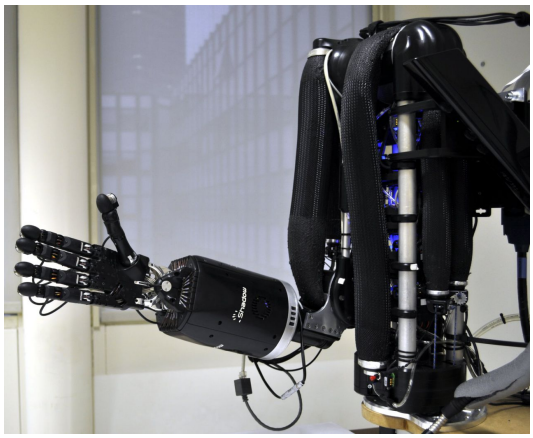


Fig. 1. *Shadow Robot platform:* The platform utilized in the test runs in Section IV-B comprises a 4 DoF arm and a hand with 20 actuated DoF.

This paper originates from efforts related to modeling/generation of grasping movements, based on a taxonomy of grasps [5], for the anthropomorphic Shadow Hand robotic platform [6], which is shown in Fig. 1. Including the two wrist joints, the hand comprises 20 controlled Degrees of Freedom (DoF). Even under consideration of possible dimensionality reduction techniques [7], [8], this requires a model capable of dealing with a substantial number of DoF. Another desideratum is the ability to incorporate multiple demonstrations since, even for the same grasp type, grasping motions can exhibit fundamentally different dynamics ($e.g.$, when starting the movement from an open and closed hand configuration). In this work we suggest an approach using a dynamical system described by Ordinary Differential Equations (ODE) to encode demonstrations provided by a user. The method incorporates the concept of Dynamical Movement Primitives (DMP) which was proposed by Ijspeert et. al. [9]. The contributions of this work are three-fold:

[1] Also commonly referred to as parameter identification, nonlinear regression or data fitting [1].

978-1-4799-2722-7/13/$31.00 

(i) In our approach, the DMP parameter estimation is formulated as a nonlinear optimization problem (instead of the usually used linear approximation) which reduces the number of parameters necessary to achieve a good fit to the provided demonstrations.

(ii) We extend the DMP concept to learning of separate DS corresponding to multiple demonstrations which allows to better capture a motion's actual underlying dynamics.

(iii) For real-time motion generation and control, we formulate a combination of the learned DS to generate a new implicit system which ensures predictable behavior over the state space. Opposed to the usage of explicit DS as in related works [9], [10], [11], [12], our formulation leaves room for online optimization and can be extended to include spatial and temporal constraints to account for additional considerations such as obstacle avoidance.

The remaining article is structured as follows: below, we briefly review related work before we introduce our DMP formulation in Section II. In Section III we suggest a method to combine multiple DS online in order to generalize over multiple demonstrations. Next, we use simulations and test runs with a robotic hand to evaluate the proposed approach. The conclusions are drawn in Section V.

### A. Related work

Dynamical systems have become a popular framework for encoding motions. In the DMP framework, the underlying DS (usually referred to as the transformation system) consists of a predefined stable linear DS which is modulated by a nonlinear forcing function that decays over time ensuring GAS. Arbitrarily many DoF can be synchronized via a phase variable (whose evolution is governed by the so called canonical system) which acts as a substitute of time. The learning problem is usually solved by fixing the nonlinear parameters of the forcing function and fitting only the linear parameters with Locally Weighted Regression (LWR). The DMP framework (see [13] for a recent review) can be used to generate point-to-point motions as well as periodic movements and lends itself well to reinforcement learning techniques [14], [15], [16]. Although DMP offer a compact way of capturing the dynamics of a single demonstration, the actual underlying dynamics can differ substantially in regions of the state space not covered by this demonstration. Hence, it is desirable to account for multiple different demonstrations to increase generalization.

Most works in this direction are based on statistical learning techniques. In [17], a statistical movement representation using Gaussian Mixture Regression is proposed. Gribovskaya et. al. [18] define a locally stable DS via a probabilistic representation of the demonstrations as a Gaussian Mixture Model (GMM). Their system is time-independent which, depending on the application, can increase robustness in the presence of temporal perturbations. Furthermore, only one DS is learned which potentially allows to capture coupling effects between different DoF. Extending the work in [18], Khansari-Zadeh et. al. [11] introduce the Stable Estimator of Dynamical Systems (SEDS) approach. Here, the parameters of the GMM are estimated by solving a Nonlinear Programming Problem (NLP). As in [18], SEDS learns a single time-independent coupled DS with additional constraints guaranteeing that the system is GAS. However, as stated by the authors in [11], with increasing number of DoF the learning problem can become intractable. Also, since the behavior of the DS in regions of the state space not covered by demonstrations depends on the specific parameters of the underlying GMM, there is no direct way of predicting the resulting state evolution.

Stulp et. al. [19] encode movement task variations, such as avoidance behaviors, using DMP learned from demonstrations with obstacles present in the demonstration scenario. In [10], Ude et. al. suggest to keep multiple demonstrated trajectories in memory and to synthesize new DMP using LWR in order to compute local models. This approach was extended in [12] to make it feasible for on-line computation by directly representing demonstrations as DMP and utilizing Gaussian Process Regression to compute new DMP parameters depending on a given desired goal point. Similarly, in [20] striking movements for table tennis are learned by mixing DMP via a gating network.

### B. Nomenclature

| | |
|---|---|
| $\tau$ | duration of a motion, |
| $M$ | number of point samples in a trajectory, |
| $D$ | number of provided demonstrations, |
| $N$ | number of basis functions parametrizing a DS, |
| $F$ | number of controlled DoF. |

Bold letters are used to denote matrices and vectors. The $k$-th element of a vector $\boldsymbol{z}$ is denoted by $z_k$.

## II. Learning dynamical movement primitives

### A. Assumptions & problem description

Our goal is to learn movement primitives by fitting the parameters of dynamical systems, described as a set of ODE with a single global attractor point, to experimental data provided in form of multiple demonstrated point-to-point trajectories in either joint- or task space. The state evolution of these dynamical systems, obtained by integrating from a given initial state, describes motion profiles which then can be converted to motor commands for the targeted platform by a low-level tracking controller. Important requirements are the ability to account for inherently different dynamics in the demonstrations and ensuring predictable behavior in regions of the state space which were not covered by the demonstrations. Also, a model structure not suffering from the curse of dimensionality is necessary, since we aim at platforms with a substantial number of DoF.

For convenience and without loss of generality, all definitions regarding dynamical systems and their respective states are stated under the assumption of an implicit change of variable, such that the equilibrium point of the considered system is at the origin [21]. A demonstrated point-to-point trajectory is given as position, velocity and acceleration vectors $\bar{\boldsymbol{q}},\ \dot{\bar{\boldsymbol{q}}},\ \ddot{\bar{\boldsymbol{q}}} \in \mathbb{R}^M$ sampled at $M$ $(m = 1, \ldots, M)$ discrete points in time. The trajectory is rescaled on a time interval between zero and one, $i.e.$, $\bar{t}_m \in [0, 1]\ \forall m$, in order to make different trajectories comparable. In accordance with the above assumption regarding the change of variable, the trajectory is shifted to converge at the origin, $i.e.$, $\bar{q}_M = 0$. For simplicity of notation we assume that each trajectory is sampled with

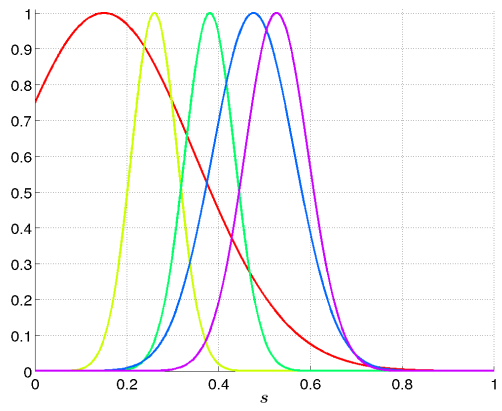



Fig. 2. *Gaussian basis functions: Shown are $N = 5$ basis functions $\Psi_n$ obtained via solving* (4) *for the demonstration in Fig. 3. The widths decrease with the distance to $s = 1$ according to the constraint $\sigma_n \leq \hat{\sigma}(1-c_n)$ in* (4)*, ensuring negligible magnitudes of $u$ for $s > 1$*

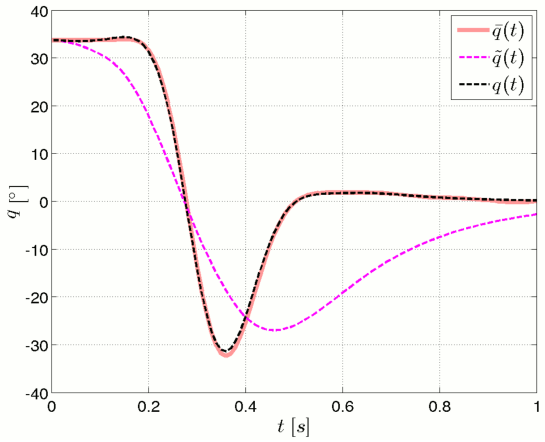



Fig. 3. *Comparison of parameter estimation methods: Shown is the reproduction ability of the DS in* (2)*, parametrized by solving* (4)*, compared to a DS using equidistantly spaced basis functions and uniform basis function widths. The result was generated by integrating the respective systems from the initial state $\bar{\boldsymbol{x}}(0)$ of the demonstration. The demonstration $\bar{q}(t)$ is denoted in pink, the dashed black line represents the position curve $q(t)$ yielded by our DS, the dashed magenta line shows the result obtained from the DS with predefined nonlinear parameters $\boldsymbol{p}$ ($\tilde{q}(t)$ was generated with the code accompanying [9]). In both cases, $N = 5$ basis functions were used.*

the same number $M$ of points, although this is not an explicit requirement of the introduced method.

### B. Encoding a single demonstration

The motion of one DoF, corresponding to a given demonstration, is encoded in a DS formulated as the ODE

$$\dot{\boldsymbol{x}} = \boldsymbol{\Phi}(\boldsymbol{x}(t), s(t); \boldsymbol{w}, \boldsymbol{p})$$

depending on parameters $\boldsymbol{w}$ and $\boldsymbol{p}$, the state $\boldsymbol{x}(t) \in \mathbb{R}^2$, and a phase variable $s(t) \in \mathbb{R}$. The phase variable provides a convenient way to scale time in order to modify the duration of the resulting motion. Its evolution is governed by the following simple dynamics

$$\frac{ds}{dt} = \dot{s} = 1/\tau, \tag{1}$$

where the scalar constant $\tau$ determines the movement's duration. The DS, together with the phase variable driving it constitutes a DMP. Synchronized motions across multiple DoF, each of which associated with a separate DS, are achieved by using a common phase variable $s(t)$. A DS consists of a linear mass-spring-damper excited by a nonlinear input $u(s) \in \mathbb{R}$ which is often referred to as a forcing function. As in [9], we choose to represent the forcing function as a weighted sum of $N$ Gaussian basis functions with weights $\boldsymbol{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, respective centers $c_n \in [0,\ 1]$ and widths $\sigma_n$ which are collected in the vector $\boldsymbol{p} = (c_1, \sigma_1, \ldots, c_N, \sigma_N) \in \mathbb{R}^{2N}$. The system $\boldsymbol{\Phi}(\boldsymbol{x}(t), s(t); \boldsymbol{w}, \boldsymbol{p})$ is given by

$$\underbrace{\begin{bmatrix}\dot{q}\\ \ddot{q}\end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix}0 & 1\\ a/\tau^2 & b/\tau\end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix}q\\ \dot{q}\end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix}0\\ 1/\tau^2\end{bmatrix}}_{\boldsymbol{B}} u(s) \tag{2}$$

$$u(s) = \sum_{n=1}^{N} \Psi_n(s; c_n, \sigma_n)\omega_n, \tag{3}$$

where the parameters $a$ and $b$ are predefined such that critical damping is enforced and $\Psi_n = \exp\left(-0.5(s-c_n)^2/\sigma_n^2\right), \forall n$. In the original DMP framework [9], the phase variable $s$ is governed by converging dynamics and used to scale the inputs $u$ in order to guarantee GAS. In our formulation this is not required since we compute the parameters of the DS by solving an optimization problem in which we enforce appropriate constraints to ensure GAS as shown in Section II-C.

To generate a motion, $s$ is reset to zero and the DS in (2) is integrated from the given initial state. When $s$ reaches one, the forcing terms $u$ become negligible. The time evolution of the phase variable, and thus the movement duration, is governed by $\tau$. Our choice of the system in (1) governing the evolution of the phase variable was made for simplicity. The use of alternative canonical systems is possible but would not qualitatively change the results.

### C. Learning DS via nonlinear programming

Learning a DMP amounts to estimating the parameters $\boldsymbol{w}$ and $\boldsymbol{p}$ of the forcing function $u(s)$ in (3). This is a nonlinear problem which is usually tackled by fixing the nonlinear parameters in $\boldsymbol{p}$ according to some heuristics (*e.g.*, uniform Gaussian widths $\sigma_n$ and equidistantly spaced centers $c_n$). Here, in a first step, we formulate a NLP in order to fit the parameters for a single system $\boldsymbol{\Phi}(\boldsymbol{x}, s; \boldsymbol{w}, \boldsymbol{p})$ to a provided demonstration. The goal is to learn forcing terms $u$ such that the system resembles the dynamics of the demonstration. This is achieved by minimizing the $L_2$ norm of the acceleration residual between the demonstrated data and the output generated by the model. The corresponding constrained nonlinear least squares problem is given below[2]

$$\underset{\boldsymbol{w},\boldsymbol{p}}{\text{minimize}} \quad \sum_{m=1}^{M} \left[\boldsymbol{C}\boldsymbol{\Phi}\left(\bar{\boldsymbol{x}}_m, \bar{s}_m; \boldsymbol{w}, \boldsymbol{p}\right) - \ddot{\bar{q}}_m\right]^2 \tag{4}$$

$$\text{subject to} \tag{5}$$

$$\begin{aligned} \sigma_n &\leq \hat{\sigma}(1-c_n), \quad n = 1, \ldots, N\\ 0 &\leq c_j \leq 1, \qquad\quad n = 1, \ldots, N\\ \Delta c_n &\leq c_n - c_{n-1}, \qquad n = 2, \ldots, N, \end{aligned}$$

[2]This problem is not convex and thus, in general, only a local minimizer will be found.

where $\bar{\boldsymbol{x}}_m = (\bar{q}_m, \dot{\bar{q}}_m)$ and $\bar{s}_m = \bar{t}_m$ due to the time scaling of the demonstrations as stated in Section II-A. $\boldsymbol{C} = [0,\ 1]$ is a selection matrix and $\Delta c : 0 \leq \Delta c \leq 1/N$ is a constant limiting the minimum distance between the centers of basis functions in order to prevent overlapping. The scalar $\epsilon : 0 < \epsilon \ll 1$ can be used to arbitrary limit the value of the basis functions at the end of the interval $s \in [0, 1]$, *i.e.*, $\Psi_n(1) \leq \epsilon, \forall n$, which ensures GAS. To this end, $\hat{\sigma} = \sqrt{-0.5/\log(\epsilon)}$ is computed as the width of a basis function centered at $c_n = 0$. To provide the solver with a feasible initial guess, the problem above is solved with fixed basis functions centers and widths which reduces (4) to a Quadratic Programming (QP) problem. Here, the $N$ initial centers $\tilde{c}_n$ are equidistantly spaced on the interval $s \in [0,\ 1]$ and the associated widths are located on the corresponding constraint in (5) such that $\tilde{\sigma}_n = \hat{\sigma}(1 - \tilde{c}_n), \forall n$.

An example of the parameters $\boldsymbol{p}$ obtained by solving (4) is shown in Fig. 2. The corresponding demonstration, along with a comparison to a solution generated with heuristically fixed nonlinear parameters is depicted in Fig. 3. Evidently, by including the nonlinear parameters $\boldsymbol{p}$ in the decision variables, a better fit can be obtained.

### D. Encoding multiple demonstrations

In the next step, the goal is to fit (for one DoF) the forcing terms of $D$ dynamical systems to $D$ provided demonstrations such that the $d$-th DS encodes the dynamics in the vicinity of the $d$-th demonstration. One could simply use the NLP in (4) to identify $\boldsymbol{w} \in \mathbb{R}^N$ and $\boldsymbol{p} \in \mathbb{R}^{2N}$ separately for each DS which would amount to estimate $3DN$ parameters. Instead, we reformulate (4) such that the nonlinear basis function parameters $\boldsymbol{p}$ are shared among the $D$ dynamical systems while the $d$-th DS has associated linear parameters $\boldsymbol{w}^d$. The objective function becomes

$$\underset{\boldsymbol{w}^1,\ldots,\boldsymbol{w}_d,\boldsymbol{p}}{\text{minimize}} \sum_{d=1}^{D} \left\{ \sum_{m=1}^{M} \left[ \boldsymbol{C}\boldsymbol{\Phi}_d\left(\bar{\boldsymbol{x}}_{d_m}, \bar{s}_m; \boldsymbol{w}_d, \boldsymbol{p}\right) - \ddot{\bar{q}}_{d_m} \right]^2 \right\} \quad (6)$$

and the problem is subjected to the constraints in (5). The above formulation allows a fit with $N(D+2)$ parameters and was used for the evaluation in Section IV.

## III. Real-time control with movement primitives

In this section we discuss how to use the previously learned DS (each of which corresponds to a demonstration) for motion generation and control. Let $\boldsymbol{x}_d[k]$ denote the state at time $t_k$ obtained by integrating $\boldsymbol{\Phi}_d(\boldsymbol{x}_d, s)$ from $t = t_1$ to $t = t_k$ starting from $\bar{\boldsymbol{x}}_d(0)$ (*i.e.*, from the initial state of the $d$-th demonstration). Our approach makes dual use of the dynamical systems. First, the set of *reference states* $\mathcal{R}[k] = \{\boldsymbol{x}_1[k], \ldots, \boldsymbol{x}_D[k]\}$ provides, at each time $t_k$, a representation of the corresponding demonstration encoded in $\boldsymbol{\Phi}_d(\boldsymbol{x}_d, s)$. Second, we formulate a movement primitive comprising a new DS where the forcing term is formed as a convex combination of individual inputs $u_d[k]$ corresponding to the systems $\boldsymbol{\Phi}_d(\boldsymbol{x}_d, s)$

$$\dot{\boldsymbol{x}}[k] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\sum_{d=1}^{D} \lambda_d[k] u_d[k]. \quad (7)$$

Here, $\boldsymbol{A}$ and $\boldsymbol{B}$ are the same as in the systems $\boldsymbol{\Phi}_d(\boldsymbol{x}_d, s)$.

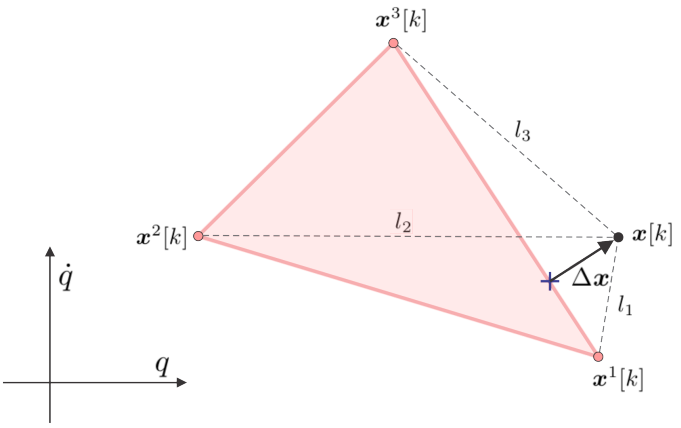



Fig. 4. *Convex combination at time $t_k$: The pink shaded area represents the convex hull over the reference states in $\mathcal{R}[k]$, the projection $\sum_{d=1}^{D} \lambda_d[k]\boldsymbol{x}_d[k]$ of the current state $\boldsymbol{x}[k]$ onto this convex hull is indicated by the blue cross, $\Delta\boldsymbol{x}$ signifies the projection residual.*

Equation (7) describes an implicit DS, where by implicit we imply that the system is not given in closed form. Rather, its definition relies on an online solution of an optimization problem. Here, the coefficients $\lambda_d[k]$ are recomputed at every time-step $t_k$ by minimizing the residual

$$\Delta\boldsymbol{x}[k] = \boldsymbol{x}[k] - \sum_{d=1}^{D} \lambda_d[k]\boldsymbol{x}_d[k] \quad (8)$$

of the projection of the current state $\boldsymbol{x}[k]$ of the actual system onto the convex hull over the current reference states $\mathcal{R}[k]$ (see Fig. 4). The associated minimization problem is stated in the QP below

$$\begin{aligned} \underset{\lambda_1[k],\ldots,\lambda_D[k]}{\text{minimize}} \quad & \|\Delta\boldsymbol{x}[k]\|_H^2 + \kappa \sum_{d=1}^{D} l_d\lambda_d[k] \\ \text{subject to} \quad & \sum_{d=1}^{D} \lambda_d[k] = 1, \\ & \lambda_d[k] \geq 0, \quad d = 1, \ldots, D, \end{aligned} \quad (9)$$

where $l_d = \|\boldsymbol{x}[k] - \boldsymbol{x}_d[k]\|_2$ and $\kappa \geq 0$ is a (small) scalar. The second term in the objective function in (9) is added in order to resolve redundancy between multiple equivalent solutions for $\lambda_d[k]$ which can occur if the residual $\Delta\boldsymbol{x}$ is zero. We define $\|\boldsymbol{z}\|_H^2 = \boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z}$ for some $\boldsymbol{z} \in \mathbb{R}^n$ and a positive semi-definite (and symmetric) matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$. Let the vector $(\lambda_1^\star, \ldots, \lambda_D^\star)$ denote a solution of (9) (*i.e.*, $\lambda_d[k] = \lambda_d^\star$). Since the coefficients $\lambda_d^\star$ are recomputed only at discrete steps $k$ according to (9), they are constant within the time window $[t_k, t_{k+1}]$.

In order to characterize the behavior of the newly formed DS in (7) we formulate the following proposition.

**Proposition 1:** *The projection residual $\Delta\boldsymbol{x}[k]$ converges onto the convex hull over the reference states $\mathcal{R}[k]$ with dynamics governed by the matrix $\boldsymbol{A}$*

$$\Delta\dot{\boldsymbol{x}}[k] = \boldsymbol{A}\Delta\boldsymbol{x}[k], \quad t \in [t_k, t_{k+1}].$$

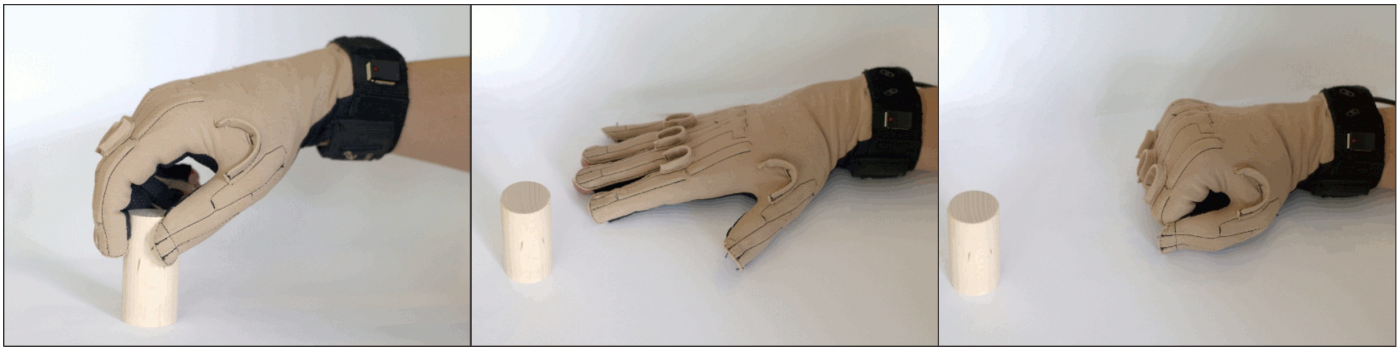


Fig. 5. *Data acquisition:* An Immersion Cyberglove-18 was used to record joint angles during grasp motions at a sample rate of 30 Hz. Starting from open and closed initial hand configurations, tripod grasps according to the taxonomy in [5] were performed on cylindrical objects.

*If the convex hull over $\mathcal{R}[k]$ contains the current state $\boldsymbol{x}[k]$, the projection residual $\Delta\boldsymbol{x}[k]$ is zero and the next state $\boldsymbol{x}[k+1]$ will be a convex combination of the reference states in $\mathcal{R}[k+1]$, i. e.,*

$$\boldsymbol{x}[k+1] = \sum_{d=1}^{D} \lambda_d^{\star} \boldsymbol{x}_d[k+1].$$

To prove the above proposition we consider for simplicity zero-order hold discretized systems, although the proof can be trivially extended to handle the continuous time case. The respective discretizations of the systems in (2) and (7) are

$$\boldsymbol{x}_d[k+1] = \bar{\boldsymbol{A}}\boldsymbol{x}_d[k] + \bar{\boldsymbol{B}}u_d[k] \tag{10}$$

$$\boldsymbol{x}[k+1] = \bar{\boldsymbol{A}}\boldsymbol{x}[k] + \bar{\boldsymbol{B}}\sum_{d=1}^{D} \lambda_d^{*} u_d[k], \tag{11}$$

where $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are the respective state transition matrix and input vector of the discrete system. Substituting (10) and (11) in (8) for time $t_{k+1}$ results in

$$\Delta\boldsymbol{x}[k+1] = \bar{\boldsymbol{A}} \underbrace{\left( \boldsymbol{x}[k] - \sum_{d=1}^{D} \lambda_d^{*} \boldsymbol{x}_d[k] \right)}_{\Delta\boldsymbol{x}[k]}, \tag{12}$$

which confirms the first part of Proposition 1.

Furthermore, we note that if the projection residual $\Delta\boldsymbol{x}[k]$ in (12) is zero, the state $\boldsymbol{x}[k]$ can be expressed as a convex combination of the reference states in $\mathcal{R}[k]$. Thus, for $\Delta\boldsymbol{x}[k] = 0$, we can rewrite (11) as

$$\begin{aligned}\boldsymbol{x}[k+1] &= \bar{\boldsymbol{A}} \underbrace{\sum_{d=1}^{D} \lambda_d^{*} \boldsymbol{x}_d[k]}_{\boldsymbol{x}[k]} + \bar{\boldsymbol{B}} \sum_{d=1}^{D} \lambda_d^{*} u_d[k] \\ &= \sum_{d=1}^{D} \lambda_d^{*} \boldsymbol{x}_d[k+1]\end{aligned}$$

which concludes the proof.

Proposition 1 summarizes the main contribution of this work. The DS in (7) accounts for different dynamics encoded from multiple demonstrations while exhibiting a predictable behavior over the whole state space. This is achieved by encoding a representation of the underlying demonstrations by means of the DS itself. States inside the convex hull of the reference states evolve according to a convex combination of the references. The matrix $\boldsymbol{A}$ in (7) governs the evolution for states outside the convex hull of the references and can be tuned according to the application. As in the original DMP formulation [9], arbitrary many DoF can be synchronized via a common phase variable $s$.

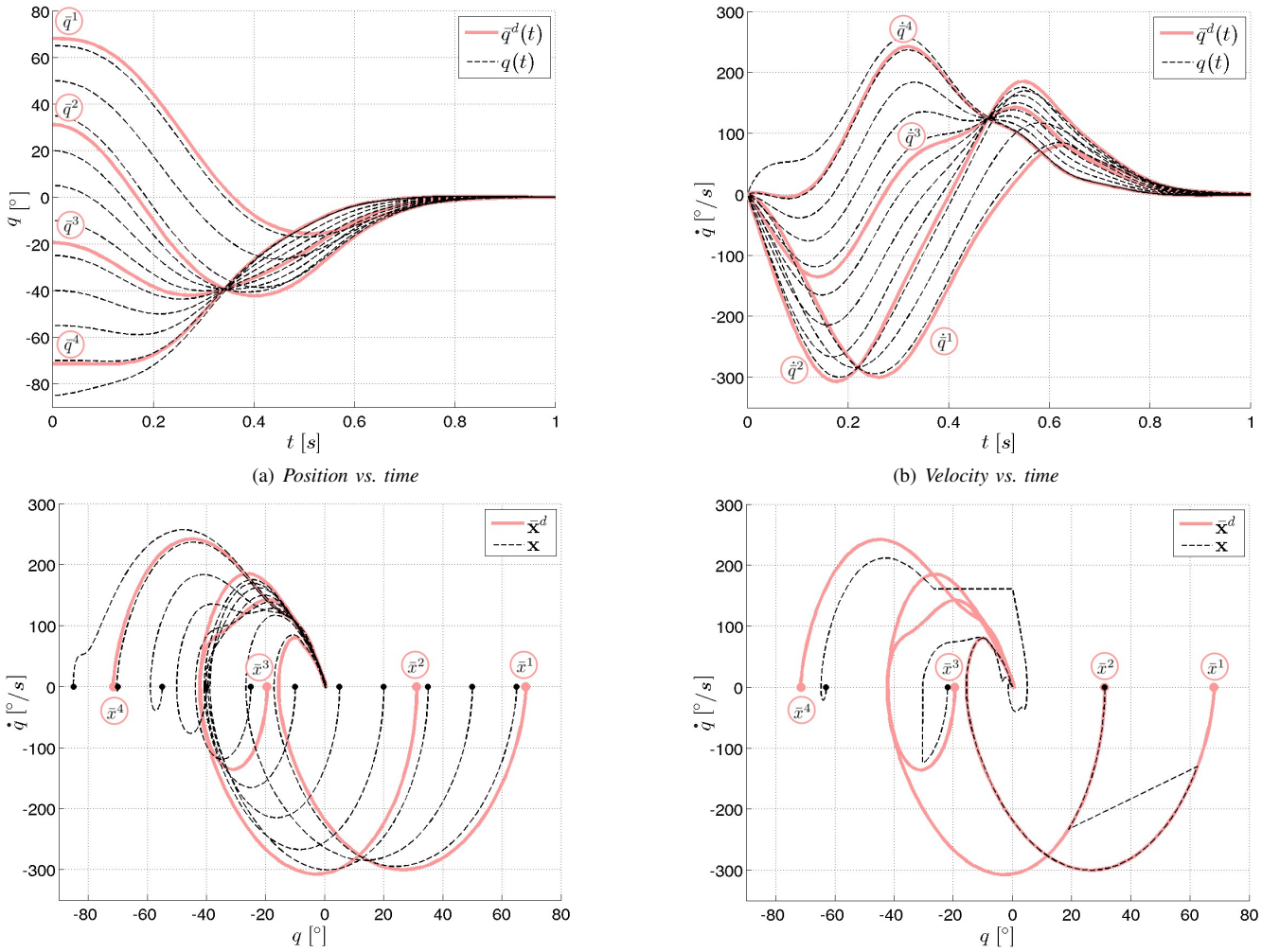
The computational load of the presented scheme at each time-step $k$ consists of integrating the canonical system in (1) and the $FD$ dynamical systems in (3), where $F$ is the number of DoF and $D$ denotes the number of DS (each corresponding to a demonstration) per DoF. Furthermore, the solution of $F$ QP's according to (9) is required.

A remaining question, which is not addressed in the scope of this paper, is how appropriate the trajectories generated by the policy in (7) are in the presence of obstacles which are not known a priori. One could imagine an example were the combination of the reference dynamics leads to collisions with unforeseen obstacles. A possible solution is to augment the underlying dynamical system in (7) with repelling potential fields such that the resulting states evolve around the obstacles as it has been shown in [22].

Opposed to existing approaches [10], [12], [20] which use statistical learning techniques to combine pre-learned DMP in order to generalize to novel situations, the suggested method provides a straightforward way to incorporate state constraints. Since the approach allows to modify the motion generating system in (7) at each time step, we currently investigate an alternative way of handling obstacles using model predictive control. To give an outlook, opposed to optimize the projection residual $\Delta\boldsymbol{x}[k]$ only w.r.t. the current time-step as in (9), we can use the linearity of the system in (7) regarding the controls $\lambda_d[k]$ to predict the behavior of the projection residual according to (8) $P$ steps forward in time to obtain

$$\underbrace{\begin{bmatrix} \Delta\boldsymbol{x}[k+1] \\ \vdots \\ \Delta\boldsymbol{x}[k+P] \end{bmatrix}}_{\Delta\boldsymbol{X}} = \begin{bmatrix} \bar{\boldsymbol{A}} \\ \vdots \\ \bar{\boldsymbol{A}}^{P-1} \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{Z} \underbrace{\begin{bmatrix} \boldsymbol{\lambda}[k] \\ \vdots \\ \boldsymbol{\lambda}[k+P-1] \end{bmatrix}}_{\boldsymbol{\Lambda}},$$

where $\boldsymbol{\lambda}[k] = (\lambda_1[k], \ldots, \lambda_D[k])$ and $\boldsymbol{Z}$ is a Toeplitz matrix (see, *e. g.*, [23]). The decision vector $\boldsymbol{\Lambda}$ is determined by minimizing $\|\Delta\boldsymbol{X}\|_H^2$ using the same prioritizing scheme as in (9) to resolve possible redundant solutions. A set of spatial and temporal polyhedral constraints designed to lead the system around given obstacles can be included in the optimization.




(a) *Position vs. time*

(b) *Velocity vs. time*

(c) *Phase plane - generalization*

(d) *Phase plane - disturbance compensation*

Fig. 6. *Generalization over demonstrations and disturbance compensation:* Black dashed lines represent the trajectories obtained by simulating the dynamical system in (7), describing the motion primitive for the MCP joint, starting from different initial conditions. The system was parametrized via the demonstrated trajectories denoted in pink. Demonstrations $d = 1$ and $d = 3$ are associated with grasps made on a cylindrical object with diameter 65 mm starting from closed and open initial hand configurations respectively, $d = 2$ and $d = 4$ correspond to grasps on an object with diameter 33 mm (see Fig. 5). (a) and (b) depict the curves for position and velocity, the corresponding phase diagram is shown in (c). The behavior of the system in the presence of disturbances is depicted in (d). After evolving unperturbed initially, the system was subjected to disturbances in position, velocity and a combined disturbance respectively.

Variants of this approach have recently been successfully applied to on-line path planning schemes for autonomous and semi-autonomous vehicles [24], [25].

## IV. Evaluation

In this section we evaluate, by means of simulations and test runs on the Shadow Robot platform, the application of the suggested method to offline learning of motion primitives from demonstrated trajectories and the usage of these primitives for real-time motion control. To this end we used a sensorized glove (see Fig. 5) to record four sets of demonstrated hand joint trajectories by performing tripod grasps on two cylindrical objects with different diameters. The recordings were made while starting from open and closed initial hand configurations respectively. The Shadow hand's joint angles were obtained via a linear regression mapping from the glove's sensor space to the robot's joint angle space. As the goal is to model grasp joint motions using DMP driven by a common phase variable $s$, the corresponding demonstrations have to live on a common time interval. Thus, all trajectories were segmented from the time a non-zero velocity was detected at a joint, until all joints stopped moving. Furthermore, the demonstrated trajectories were smoothed by means of a linear least squares regression and numerically differentiated to obtain velocities and accelerations. After rescaling and shifting, as described in Section II-A, the trajectories were re-sampled with a number of $M = 100$ points each.

TABLE I. Fixed parameters used in the evaluation

| $N$ | $a$ | $b$ | $\epsilon$ | $\Delta c$ | $\boldsymbol{H}$ | $\kappa$ |
|---|---|---|---|---|---|---|
| 5 | $-132.5$ | $-23$ | $10^{-4}$ | 0.05 | diag(100, 1) | 0 |

As described in Section II-D, for the $F = 20$ DoF of the Shadow hand we used the demonstrated trajectories to estimate the free parameters of 20 motion primitives according to (6), the utilized fixed parameters are summarized in Table I. The constrained nonlinear least squares problems in (6) were solved with a Sequential Quadratic Programming (SQP) algorithm, utilizing the ACADO Toolkit [26].

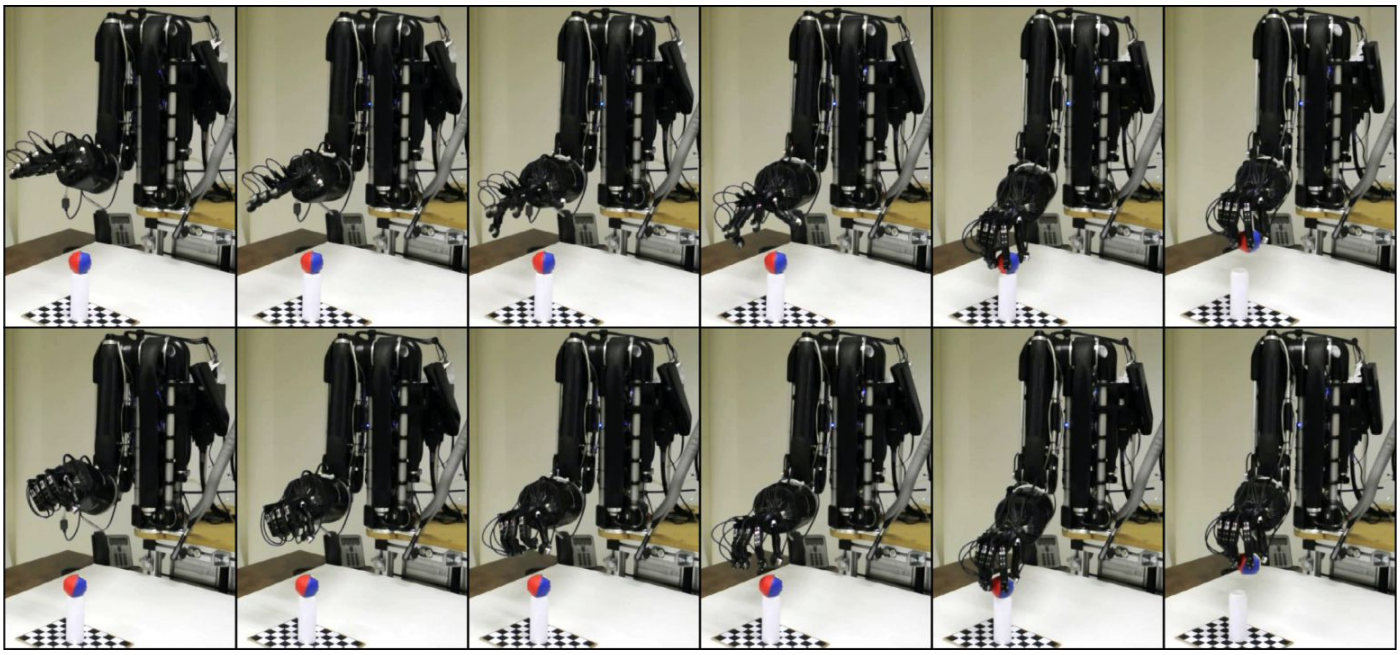


Fig. 7. *Tripod grasp primitives triggered from different initial configurations:* Synchronized finger joint movements are generated by means of integrating motion primitives corresponding to (7) which are driven by a common phase variable. Top row: Starting from an open hand configuration; bottom row: starting from a closed hand configuration.

### A. Simulated experiments

To assess the generalization capabilities of the learned models for the considered point-to-point movements, we performed simulations by initializing the motion primitives from different initial states. Exemplary, the results for the dynamical system describing the flexion/extension motions of the middle fingers Metacarpophalangeal (MCP) joint (the MCP joints connect the proximal phalanges of the fingers to the palm) are shown in Fig. 6. Depicted are the obtained position, velocity and phase plane curves. As argued in Section III, for states evolving inside the convex hull over the reference states the distance ratio to the references is governed by the convex combination coefficients computed as a solution of (9). States outside the convex hull over the references are attracted towards this convex hull according to dynamics governed by the matrix $\boldsymbol{A}$ in (7). It can be seen that the model can reproduce the demonstrated trajectories with high fidelity while exhibiting a deterministic behavior in regions of the state space not covered by the demonstrations.

Furthermore, we investigated the behavior of the model in the presence of state disturbances. We investigated separate position and velocity disturbances as well as a combined disturbance. When, at time $t_k$, the system is perturbed inside the convex hull of the reference states, the update of the convex combination coefficients according to (9) at time $t_{k+1}$ adjusts the future evolution of the system according to the reference states at time $t_{k+1}$. An example is shown in Fig. 6(d) where a trajectory was started at the initial state $\bar{\boldsymbol{x}}^2(0)$ corresponding to the second demonstration and is pushed onto the reference trajectory associated with the first demonstration. After adjusting the combination coefficients in the next time step, the system continues to evolve according to $\bar{\boldsymbol{x}}^1$. Disturbances with states resulting outside the convex hull of the references again cause the system to converge towards the projection onto this convex hull with dynamics as specified in (7).

### B. Test runs on the Shadow hand

Here, the goal is to ascertain the feasibility of the developed motion primitives for real-time motion generation and control rather then to show a fully applicable grasping/manipulation system for which other components such as grasp planning, object perception and obstacle avoidance are necessary which are not in the scope of this work. A standard laptop was used to control the Shadow Robot platform via the Robot Operating System (ROS) framework at 100 Hz. The learned motion primitives were used to generate motion profiles for the 20 DoF of the Shadow hand. Appropriate motion profiles for the 4 DoF of the arm were generated with the ROS joint spline trajectory controllers, such that hand and arm motion comprised the same duration. A desired final hand/arm configuration for a tripod grasp on the ball in Fig. 7 was obtained via kinesthetic teaching and subsequently adding an empiric small increment to the joint values in order to ensure sufficient squeezing of the object. Then, the motion primitives for the hand joints were triggered from initial conditions corresponding to open, pronated and closed hand configurations respectively which allowed to successfully execute synchronized grasp and subsequent lifting motions as shown in Fig. 7. Here, the arm joints were moved between predefined start- and final positions. One encountered problem was that the ROS messaging system introduced unacceptable feedback delays and that the available low-level position PID tracking control was of limited quality. Thus, the test runs were carried out in an open-loop fashion, $i.e.$, the primitives were only used for online planning of reference profiles between the given start and end positions without considering state feedback. Figure 8 shows, again exemplary for the MCP joint, the controller set points obtained from integrating the DS and the resulting position curves generated by the tracking controller. Despite the obvious limitations in the low-level control, the grasping tasks were conducted successfully.

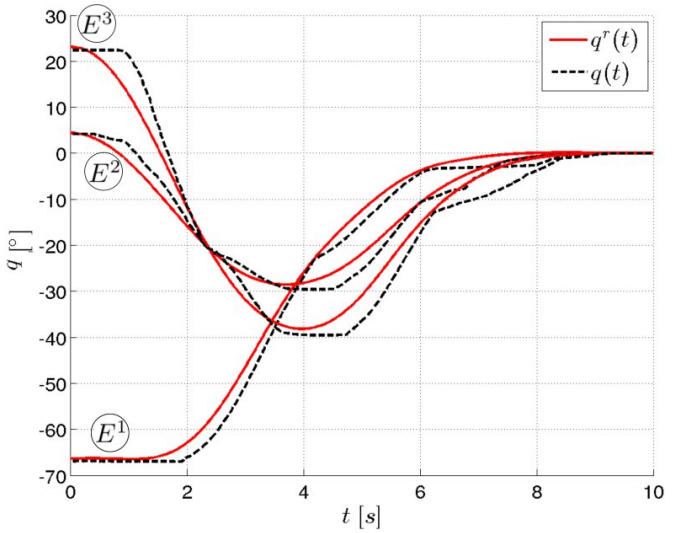



Fig. 8. *Test runs on Shadow hand:* Tracking controller set-points $q^r(t)$ and position curves $q(t)$ for the MCP joint starting from open (experiment $E^1$), pronated ($E^2$) and closed ($E^3$) initial hand configuration. Motion duration $\tau = 10s$.

## V. Conclusions

In this work we present an approach using demonstrated motion data in order to parametrize dynamical systems for movement generation via nonlinear optimization. Offline learning is used to fit the parameters of dynamical systems to the demonstrated data. For real-time control, we form a new implicit system as a locally optimal combination of the previously learned DS. This results in a deterministic behavior in state regions which were not explored during the demonstrations. Furthermore, the demonstrations can be reproduced with high fidelity while relying on a comparatively small number of parameters. We assessed the introduced method by means of parametrizing the proposed model from demonstrations of grasp movements and subsequent simulations and test runs with the Shadow Robot platform. Our approach affords the flexibility to modify the control inputs of the implicit system used for motion generation at each time-step. Consequently, future work will include the incorporation of spatial and temporal state space constraints for obstacle avoidance in order to realize a reactive on-line planning/control scheme.

## Acknowledgments

This research has been partially supported by the projects HANDLE (grant agreement ICT-231640) and ROBLOG (grant agreement ICT-270350), funded by the European Community's Seventh Framework Program (FP7/2007-2013). The authors would like to thank Guillaume Walck at ISIR, UPMC Paris for his support with the Shadow Robot platform.

## References

[1] K. Schittkowski, *Numerical Data Fitting in Dynamical Systems*. Kluwer Academic Publishers, 2002.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 1371 – 1394.

[3] J.-H. Hwang, R. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proc. of the IEEE/RSJ Int. Conf. on Int. Robots and Systems*, vol. 2, 2003, pp. 1444 – 1449.

[4] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 409 – 413, 2006.

[5] T. Feix, R. Pawlik, H. Schmiedmayer, J. Romero, and D. Kragic, "A comprehensive grasp taxonomy," in *RSS: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation*, 2009.

[6] Shadow Robot Company, "The shadow dextrous hand." [Online]. Available: http://www.shadowrobot.com/hand/

[7] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *IJRR*, vol. 28, no. 7, pp. 851 – 867, 2009.

[8] M. Gabiccini, A. Bicchi, D. Prattichizzo, and M. Malvezzi, "On the role of hand synergies in the optimal choice of grasping forces," *Autonomous Robots*, vol. 31, pp. 235 – 252, 2011.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems*. MIT Press, 2003, pp. 1523 – 1530.

[10] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800 – 815, 2010.

[11] S. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943 – 957, 2011.

[12] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1327 – 1339, 2012.

[13] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, pp. 328 – 373, 2013.

[14] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *11th IEEE-RAS Int. Conf. on Humanoid Robots*, 2011, pp. 231 – 238.

[15] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 5703 – 5708.

[16] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1 - 2, pp. 171 – 203, 2011.

[17] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[18] E. Gribovskaya, S. M. Khansari-Zadeh, and A. Billard, "Learning nonlinear multivariate dynamics of motion in robotic manipulators," *IJRR*, vol. 30, no. 1, pp. 80 – 117, 2011.

[19] F. Stulp, E. Oztop, P. Pastor, M. Beetz, and S. Schaal, "Compact models of motor primitive variations for predictable reaching and obstacle avoidance," in *IEEE-RAS International Conference on Humanoid Robots*, 2009, pp. 589 – 595.

[20] K. Muelling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *IIJR*, no. 3, pp. 263 – 279, 2013.

[21] H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[22] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 2587 – 2592.

[23] J. M. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[24] F. Pecora, M. Cirillo, and D. Dimitrov, "On mission-dependent coordination of multiple vehicles under spatial and temporal constraints," in *Proc. of the IEEE/RSJ Int. Conf. on Int. Robots and Systems*, 2012, pp. 5262– 5269.

[25] S. Anderson, S. Karumanchi, and K. Iagnemma, "Constraint-based planning and control for safe, semi-autonomous operation of vehicles," in *IEEE Intelligent Vehicles Symposium*, 2012, pp. 383 – 388.

[26] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – an open source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298 – 312, 2011.