Noname manuscript No.
(will be inserted by the editor)

Model Predictive Motion Control based on Generalized Dynamical Movement Primitives

Robert Krug · Dimitar Dimitrov

the date of receipt and acceptance should be inserted later

Abstract In this work, experimental data is used to estimate the free parameters of dynamical systems intended to model motion profiles for a robotic system. The corresponding regression problem is formed as a constrained non-linear least squares problem. In our method, motions are generated via embedded optimization by combining dynamical movement primitives in a locally optimal way at each time step. Based on this concept, we introduce a model predictive control scheme which allows generalization over multiple encoded behaviors depending on the current position in the state space, while leveraging the ability to explicitly account for state constraints to the fulfillment of additional tasks such as obstacle avoidance. We present a numerical evaluation of our approach and a preliminary verification by generating grasping motions for the anthropomorphic Shadow Robot hand/arm platform.

Keywords Motion Control · Model Predictive Control · Motion Planning · Imitation Learning · Grasping

Mathematics Subject Classification (2010) 70E60 · 68T40 · 68T05 · 93C10 · 49M99

1 Introduction

On the frontier between motion planning and control, Dynamical Systems (DS) have emerged as a popular way to encode desired movement behaviors in form of state transition policies. Here, opposed to strictly following pre-planned paths or

Robert Krug
AASS Research Center, Örebro University, Sweden
Tel.: +46-19-302143
Fax: +46-19-303463
E-mail: robert.krug@oru.se

Dimitar Dimitrov
INRIA Rhône-Alpes, 38331 St Ismier Cedex, France
E-mail: dimitar.dimitrov@inria.fr

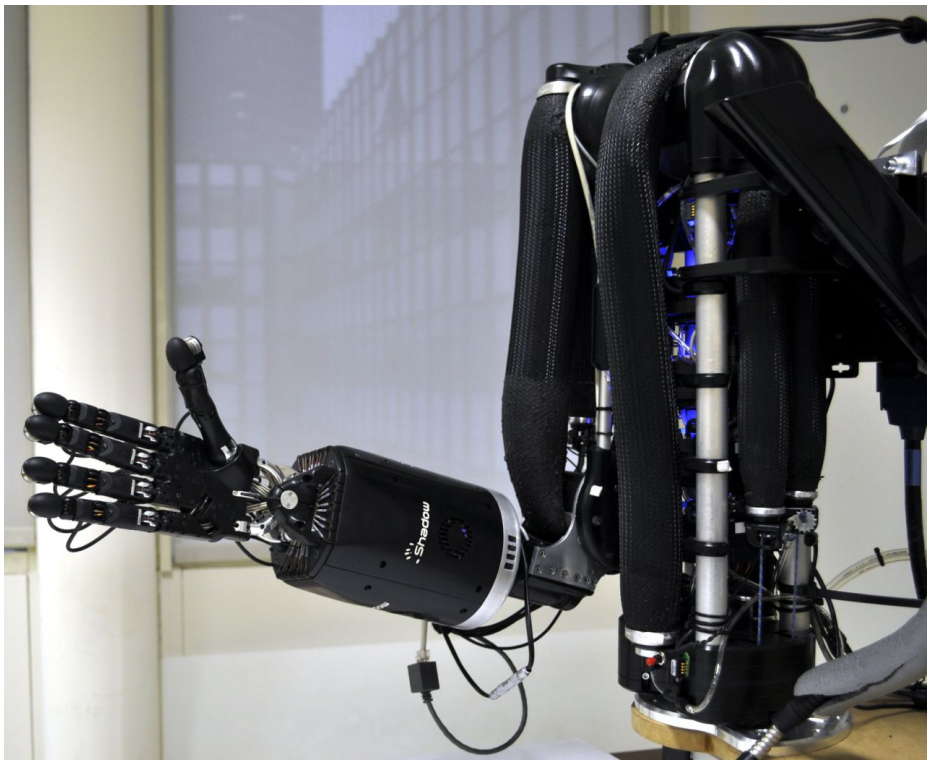


Fig. 1 *The Shadow Robot platform at ISIR, UPMC Paris:* The platform utilized in the test runs in Section 6.2 comprises a 4 DoF arm and a hand with 20 actuated DoF. Five ATI-Nano17 6D force/torque sensors embedded in the fingertips enable tactile sensing.

using spline-based methods [1, 2], motions are generated reactively which provides robustness to perturbations occurring during execution.

In order to generate "appropriate" motion patterns for a targeted robotic system, the underlying DS parameter estimation problem[1] is commonly solved by providing data examples specifying desired transitions from given initial to final states. One way to provide experimental data is to record movements of a human expert in a Programming by Demonstration setting [4]. Another possibility is to create data artificially, *e. g.*, in form of smooth minimum-jerk trajectories [5] or as the pre-computed solutions of optimal control problems [6].

The choice of an appropriate DS for motion generation is typically guided by the ability of the underlying model to generalize over the provided examples 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. Especially in an imitation learning setting where the provided demonstrations are usually relatively sparse, 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

[1] Also referred to as parameter identification, nonlinear regression or data fitting [3].

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.

Since the considered DS constitute policies over the state space whose state evolution is guaranteed to converge, they can be seen as global planners which always reach their goal in the absence of obstacles [7]. In the context of reactive planning schemes, obstacles are typically dealt with locally - often by modeling them with repelling potential fields as suggested by Khatib [8].

The presented work originates from efforts related to modeling and generation of grasping movements, based on demonstrations of taxonomic grasps [9], for the anthropomorphic Shadow Hand robotic platform [10] 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 [11], 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. [12]. The contributions of this work are the following:

(i) 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. The corresponding parameter estimation is carried out using nonlinear optimization (instead of the usually used linear approximation) which reduces the number of parameters necessary to achieve a good fit to the provided demonstrations.

(ii) For real-time motion generation and control, we employ online optimization and introduce a linear receding horizon Model Predictive Control (MPC) scheme, which is based on a convex combination of the learned DS ensuring predictable behavior over the state space. Opposed to the usage of explicit DS as in related works [12–15], our formulation is able to account for spatial and temporal constraints to account for additional considerations such as obstacle avoidance.

Part of this work has been published in preliminary form in [16]. Here, we give a more extensive numerical evaluation of our DMP learning method and extend our previous online optimization approach to a MPC scheme. The remaining article is structured as follows: below, we review related work before we formalize the tackled problem in Section 3. Our DMP formulation is introduced in Section 4. In Section 5 we suggest a method to combine multiple DS online in order to generalize over multiple demonstrations and introduce our MPC scheme for obstacle avoidance. Next, we use simulations and test runs with a robotic hand to evaluate the proposed approach in Section 6 before we draw our conclusions in Section 7.

2 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 [17] 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 [18–22]. 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 aiming at generalization of DMP are based on statistical learning techniques. Pastor et. al. [23] build a library of template primitives which can be used for sequencing movements. Matsubara et. al. [24] learn DMP from multiple demonstrations and combine them using a style parameter. In [25], a statistical movement representation using Gaussian Mixture Regression is proposed. Ude et. al. [13] suggest to keep multiple demonstrated trajectories in memory and to synthesize new primitives using LWR in order to compute local models. This approach was extended in [15] 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 [26] striking movements for table tennis are learned by mixing primitives via a gating network.

An alternative DS model structure was proposed by Gribovskaya et. al. [27]. Here, the authors 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 [27], Khansari-Zadeh et. al. [14] 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 [27], SEDS learns a single time-independent coupled DS with additional constraints guaranteeing that the system is GAS. However, as stated by the authors in [14], 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.

In a reactive planning setting based on DS, obstacles are typically dealt with locally by augmenting the DS formulation with repelling potential fields [8,28]. Alternatives include the use of coupling feed-forward terms [29] and appropriate modulation of the original DS depending on the distance of the current state to the obstacles [7,30]. With increasing maturity of online optimization algorithms

and solvers, it is becoming feasible to formulate obstacles directly as constraints in the state space [31,32]. Approaches in this mould require online solution of optimization problems during motion execution, in order to ensure that the constraints are obeyed at each point in time. Variants of this concept have recently been successfully applied to on-line path planning schemes for autonomous/semi-autonomous vehicles [33,34] and multi-robot production systems [35]. In a similar work, Da Silva [36] use MPC to track demonstrated human motions in simulation.

3 Problem Description and Assumptions

Nomenclature

	Indices
m	Trajectory point index, $m \in \{1, \dots, M\}$
d	Demonstration index, $d \in \{1, \dots, D\}$
n	Gaussian basis function index, $n \in \{1, \dots, N\}$
f	DoF index, $f \in \{1, \dots, F\}$
p	Preview window index, $p \in \{1, \dots, P\}$
h	Hyperplane index, $h \in \{1, \dots, H\}$
k	Discrete time index, $k \in \mathbb{Z}_+$
	General
$\boldsymbol{q}$	Joint configuration, $\boldsymbol{q} = [q_1, \dots, q_F]^T$
$\boldsymbol{x}$	State vector, $\boldsymbol{x} = [q,\, \dot{q}]^T$
$\bar{\boldsymbol{q}}$	Discretized demonstration, $\bar{\boldsymbol{q}} = [\bar{q}_1, \dots, \bar{q}_M]^T$
$\bar{t}$	Dilated time, $\bar{t} \in [0,\, 1]$
$\boldsymbol{\Phi}(\cdot)$	Dynamical Movement Primitive, $\boldsymbol{\Phi} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$
s	Phase variable, $s \in \mathbb{R}$
$u(\cdot)$	Forcing function, $u : \mathbb{R} \to \mathbb{R}$
$\Psi_n(\cdot)$	n-th GBF, $\Psi_n : \mathbb{R} \to \mathbb{R}$
$\boldsymbol{w}$	GBF weights, $\boldsymbol{w} = [w_1, \dots, w_N]^T$
$\boldsymbol{p}$	GBF centers and widths, $\boldsymbol{p} = [c_1, \sigma_1, \dots, c_N, \sigma_N]^T$
ϵ	Basis function limit at $s = 1$, $\epsilon \in \mathbb{R}_+$
τ	Motion duration, $\tau \in \mathbb{R}_+$
$\boldsymbol{A}$	Continuous system matrix, $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$
$\boldsymbol{B}$	Continuous input matrix, $\boldsymbol{B} \in \mathbb{R}^2$
$\bar{\boldsymbol{A}}$	Discrete state transition matrix, $\bar{\boldsymbol{A}} \in \mathbb{R}^{2\times 2}$
$\bar{\boldsymbol{B}}$	Discrete control matrix, $\bar{\boldsymbol{B}} \in \mathbb{R}^2$
$\kappa,\ \nu$	Penalty coefficients, $\kappa \in \mathbb{R}_+$, $\nu \in \mathbb{R}_+$
$\boldsymbol{C}$	Selection matrix, $\boldsymbol{C} \in \mathbb{R}^2$
$(\boldsymbol{H},\ \boldsymbol{e})$	State constraints, $\boldsymbol{H} \in \mathbb{R}^{C\times F}$, $\boldsymbol{e} \in \mathbb{R}^C$

Our goal is to develop a reactive motion generation system whose output trajectories resemble given demonstrations and which allows to incorporate state constraints for auxiliary targets such as obstacle avoidance. To this end, we 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 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 [37]. 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 discrete points in time. The trajectory is rescaled on a time interval between zero and one, *i.e.*, $\bar{t}_m \in [0,1],\ m = 1,\dots,M$, 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 the same number M of points and that the same number D of demonstrations is provided for each DoF, although these are not explicit requirements of the proposed methods. Although we present our approach for motion generation in configuration space, it is equally applicable in task space.

4 Learning Dynamical Movement Primitives

In this Section we revisit the DMP learning approach described in [16] and first show, for one DoF, how to learn a motion primitive from a single demonstration by solving a NLP. Subsequently, we extend the formulation to account for multiple demonstrations which allows to encode fundamentally different dynamics for the same DoF.

4.1 Encoding a Single Demonstration

The motion of one DoF, corresponding to a given demonstration, is encoded in a DS $\boldsymbol{\Phi} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ formulated as the ODE

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

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 $\tau \in \mathbb{R}_+$ 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 is associated with a separate

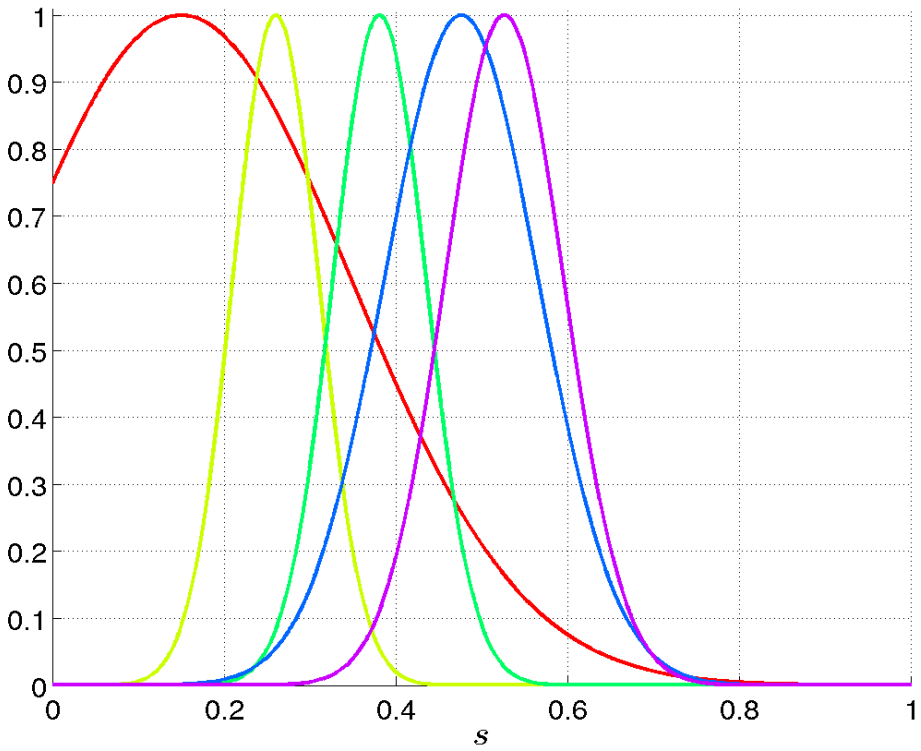


Fig. 2 *Gaussian basis functions: Shown are $N = 5$ basis functions Ψ_n obtained via solving (4) for the demonstration in Fig. 3. The widths decrease with the distance to $s = 1$ according to the constraint $\sigma_n \leq \hat{\sigma}(1 - c_n)$ in (4), ensuring negligible magnitudes of u for $s > 1$*

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 $u(s) : \mathbb{R} \to \mathbb{R}$ which is often referred to as a forcing function. As in [12], we choose to represent the forcing function as a weighted sum of N Gaussian Basis Functions (GBF) with weights $\boldsymbol{w} = [w_1, \ldots, w_N]^T \in \mathbb{R}^N$, respective centers $c_n \in [0,\ 1]$ and widths σ_n which are collected in the vector $\boldsymbol{p} = [c_1, \sigma_1, \ldots, c_N, \sigma_N]^T \in \mathbb{R}^{2N}$. The system $\boldsymbol{\Phi}(\boldsymbol{x}(t), s(t); \boldsymbol{w}, \boldsymbol{p})$ is given by

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

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

where $\alpha \in \mathbb{R}_-$ and $\beta \in \mathbb{R}_-$ are predefined such that critical damping is enforced and $\Psi_n = \exp\left(-0.5(s - c_n)^2/\sigma_n^2\right)$. In the original DMP framework [12], 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 4.2.

To generate a motion, s is reset to zero and the DS in (2) is integrated from a 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 τ. 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.

4.2 Parameter Estimation via Nonlinear Programming

Learning a DMP amounts to estimating the GBF 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 σ_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 squared 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]

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

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

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

where $\bar{\boldsymbol{x}}_m = [\bar{q}_m,\, \dot{\bar{q}}_m]^T$ and $\bar{s}_m = \bar{t}_m$ due to the time scaling of the demonstrations as stated in Section 3. $\boldsymbol{C} = [0,\, 1]$ is a selection matrix and $\Delta c \in \mathbb{R}$, $0 \leq \Delta c \leq 1/N$ is a constant limiting the minimum distance between the centers of basis functions in order to prevent overlapping. The scalar $\epsilon \in \mathbb{R}$, $0 < \epsilon \ll 1$ 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, $\hat{\sigma} = \sqrt{-0.5/\log(\epsilon)}$ corresponds to 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 as shown in Section 6.1.

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

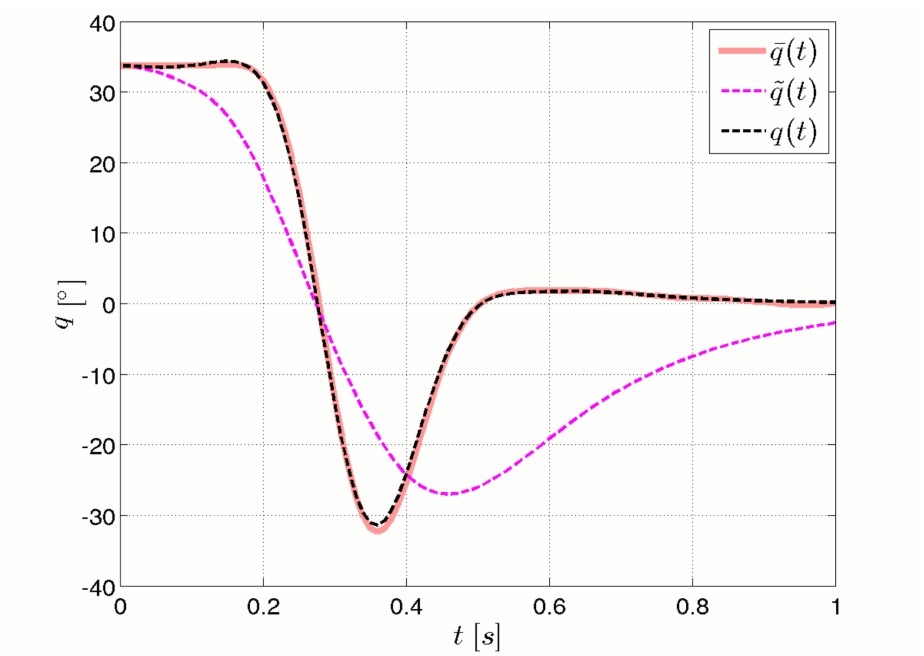


Fig. 3 *Comparison of parameter estimation methods: Shown is the reproduction ability of the DS in* (2)*, parametrized by solving* (4)*, compared to a DS using equidistantly spaced basis functions and uniform basis function widths. The result was generated by integrating the respective systems from the 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 [12]). In both cases,* $N = 5$ *basis functions were used.*

4.3 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

$$\underset{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_D,\boldsymbol{p}}{\text{minimize}} \frac{1}{2} \sum_{d=1}^{D} \left\{ \sum_{m=1}^{M} \left[\boldsymbol{C}\boldsymbol{\Phi}_d \left(\bar{\boldsymbol{x}}_{d,m}, \bar{s}_m; \boldsymbol{w}_d, \boldsymbol{p} \right) - \ddot{\bar{q}}_{d,m} \right]^2 \right\} \tag{6}$$

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 6. The concept of sharing basis functions between motion generators is similar as the one used by Rückert and d'Avella in [38], where it is put in the context of muscular synergies.

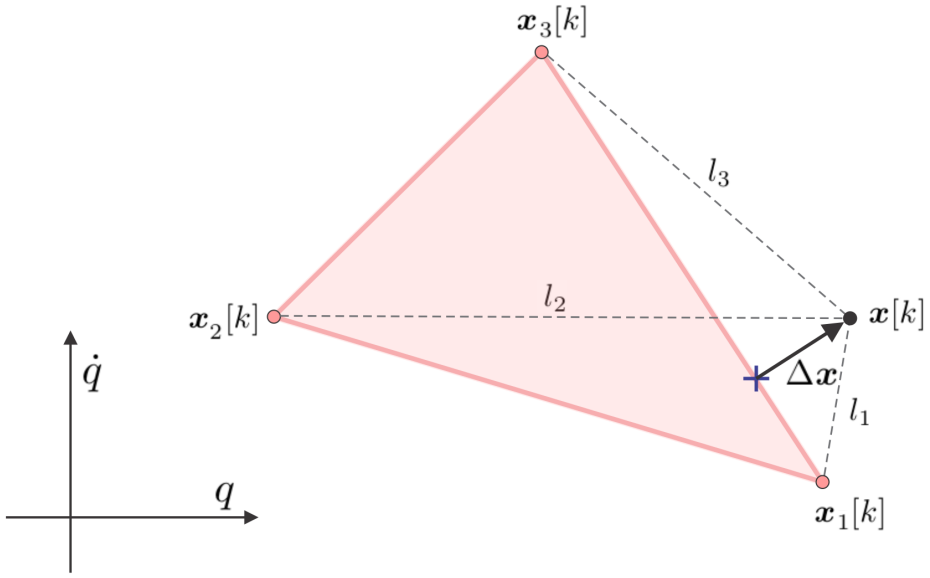


Fig. 4 *Convex combination at time t_k: The pink shaded area represents the convex hull over the reference states in $\boldsymbol{R}[k]$, the projection $\boldsymbol{R}[k]\boldsymbol{\lambda}[k]$ of the current state $\boldsymbol{x}[k]$ onto this convex hull is indicated by the blue cross, $\Delta\boldsymbol{x}$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

5 Real-time Control with Movement Primitives

In this section we first discuss how to form a new implicit DS based on a locally optimal combination of the previously learned systems (each of which corresponds to a demonstration). Then, we proceed to derive our MPC scheme with state constraints.

5.1 Generating Locally Optimal Motions

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* collected in the columns of the matrix

$$\boldsymbol{R}[k] = [\boldsymbol{x}_1[k], \ldots, \boldsymbol{x}_d[k]] \in \mathbb{R}^{2\times D} \tag{7}$$

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 $\boldsymbol{\Phi}_d(\boldsymbol{x}_d, s)$

$$\dot{\boldsymbol{x}}[k] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k]^T\boldsymbol{\lambda}[k], \tag{8}$$

where $\boldsymbol{u}[k] = [u_1[k], \ldots, u_D[k]]^T$ and $\boldsymbol{\lambda}[k] = [\lambda_1[k], \ldots, \lambda_D[k]]^T$. Here, $\boldsymbol{A}$ and $\boldsymbol{B}$ are the same as in (2). Equation (8) 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

$$\Delta\boldsymbol{x}[k] = \boldsymbol{x}[k] - \boldsymbol{R}[k]\boldsymbol{\lambda}[k] \tag{9}$$

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

$$\begin{aligned} \underset{\boldsymbol{\lambda}[k]}{\text{minimize}} \quad & \|\Delta\boldsymbol{x}[k]\|_H^2 + \kappa\boldsymbol{l}[k]^T\boldsymbol{\lambda}[k] \\ \text{subject to} \quad & \mathbb{1}^T\boldsymbol{\lambda}[k] = 1, \\ & \boldsymbol{\lambda}[k] \geq \mathbf{0}, \end{aligned} \tag{10}$$

where the elements $l_d[k] = \|\boldsymbol{x}[k] - \boldsymbol{x}_d[k]\|_2$ of the vector $\boldsymbol{l}[k] = [l_1[k], \ldots, l_D[k]]^T$ describe the euclidean distances of the reference states to the current states, $\kappa \geq 0$ is a (small) scalar and $\mathbb{1}$ is an appropriately dimensioned column vector of ones. The second term in the objective function in (10) is added in order to resolve the redundancy between multiple equivalent solutions for $\boldsymbol{\lambda}[k]$ 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 $\boldsymbol{z} \in \mathbb{R}^Z$ and a positive semi-definite (and symmetric) matrix $\boldsymbol{H} \in \mathbb{R}^{Z\times Z}$. Let the vector $\boldsymbol{\lambda}^\star = [\lambda_1^\star, \ldots, \lambda_D^\star]^T$ denote a solution of (10) (*i.e.*, $\boldsymbol{\lambda}[k] = \boldsymbol{\lambda}^\star$). The coefficients $\lambda_d^\star$ are recomputed only at discrete steps k according to (10) and are assumed to be constant within the time window $[t_k, t_{k+1}]$.

In order to characterize the behavior of the newly formed DS in (8) we formulate the following proposition.

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

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

If the convex hull over the states in $\boldsymbol{R}[k]$ 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 $\boldsymbol{R}[k+1]$, i. e.,

$$\boldsymbol{x}[k+1] = \boldsymbol{R}[k+1]\boldsymbol{\lambda}^\star.$$

A proof of the above proposition is given in Appendix A. Proposition 1 summarizes a key concept in this work. The DS in (8) 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 (8) 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 [12], arbitrary many DoF can be synchronized via a common phase variable s. What sets this work conceptually apart from existing approaches such as presented in [23,13–15,6], is the ability to modify the dynamical system which drives the motion on the fly via embedded optimization. This is a useful ability in the context of, *e. g.*, obstacle avoidance and disturbance compensation as shown in the following.

5.2 DMP-based Model Predictive Control

A remaining question is how appropriate the trajectories generated by the policy in (8) 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.

Opposed to existing approaches [24, 13, 15, 26] 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 (8) at each time step, we suggest an alternative way of handling obstacles using model predictive control under a set of spatial and temporal polyhedral constraints which are designed to lead the system around a given (potentially moving) obstacle.

To start, let us note that the matrix formed by the product $\boldsymbol{B}\boldsymbol{u}[k] \in \mathbb{R}^{2\times D}$ in (8) can loose rank (*e. g.*, towards the end of a motion when the elements of $\boldsymbol{u}[k]$ vanish) and that the vector $\boldsymbol{\lambda}[k]$ is bound by the convex constraints in (10). Therefore, to ensure the ability to satisfy additional state constraints, we augment the system in (8) with an auxiliary control input $\tilde{\lambda}$ and discretize to obtain

$$\boldsymbol{x}[k+1] = \bar{\boldsymbol{A}}\boldsymbol{x}[k] + \bar{\boldsymbol{B}}\boldsymbol{\mu}[k]^T\boldsymbol{\gamma}[k], \tag{11}$$

where $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are state transition matrix and control matrix of the discrete system, $\boldsymbol{\mu}[k] = [\boldsymbol{u}[k], 1]^T \in \mathbb{R}^{D+1}$ and $\boldsymbol{\gamma}[k] = \left[\boldsymbol{\lambda}[k], \tilde{\lambda}[k]\right]^T \in \mathbb{R}^{D+1}$ denotes the augmented control vector. Next, we want to predict the residual of the projection of the current state on the reference states P step forwards in time. Inserting the augmented system in (11) in the residual formulation in (9) and performing recursion yields

$$\underbrace{\begin{bmatrix} \Delta\boldsymbol{x}[k] \\ \Delta\boldsymbol{x}[k+1] \\ \vdots \\ \Delta\boldsymbol{x}[k+P] \end{bmatrix}}_{\Delta\boldsymbol{X} \in \mathbb{R}^{2(P+1)}} = \begin{bmatrix} \bar{\boldsymbol{A}}^0 \\ \bar{\boldsymbol{A}}^1 \\ \vdots \\ \bar{\boldsymbol{A}}^P \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{Z} \underbrace{\begin{bmatrix} \boldsymbol{\gamma}[k] \\ \boldsymbol{\gamma}[k+1] \\ \vdots \\ \boldsymbol{\gamma}[k+P] \end{bmatrix}}_{\boldsymbol{\Gamma} \in \mathbb{R}^{(D+1)(P+1)}}, \tag{12}$$

where the matrix $\boldsymbol{Z} \in \mathbb{R}^{2(P+1)\times(D+1)(P+1)}$ is given as

$$\boldsymbol{Z} = \begin{bmatrix} -\boldsymbol{R}[k] & \boldsymbol{0} & \dots & \dots & \boldsymbol{0} \\ \bar{\boldsymbol{B}}\boldsymbol{\mu}[k] & -\boldsymbol{R}[k+1] & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \bar{\boldsymbol{A}}\bar{\boldsymbol{B}}\boldsymbol{\mu}[k] & \bar{\boldsymbol{B}}\boldsymbol{\mu}[k+1] & -\boldsymbol{R}[k+2] & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \boldsymbol{0} \\ \bar{\boldsymbol{A}}^{P-1}\bar{\boldsymbol{B}}\boldsymbol{\mu}[k] & \bar{\boldsymbol{A}}^{P-2}\bar{\boldsymbol{B}}\boldsymbol{\mu}[k+1] & \dots & \bar{\boldsymbol{B}}\boldsymbol{\mu}[k+P-1] & -\boldsymbol{R}[k+P] \end{bmatrix}.$$

Without consideration of additional state constraints, we can now formulate a receding horizon MPC scheme as the following optimization problem which needs

to be solved at every time step t_k

$$\begin{aligned} \underset{\boldsymbol{\Gamma}}{\text{minimize}} \quad & \|\Delta \boldsymbol{X}[k]\|_H^2 + \kappa \boldsymbol{l}[k]^T \boldsymbol{\lambda}[k] + \nu \sum_{p=0}^{P} \tilde{\lambda}[k+p]^2 \\ \text{subject to} \quad & \mathbb{1}^T \boldsymbol{\lambda}[k+p] = 1, \quad p = 0, \dots, P, \\ & \boldsymbol{\lambda}[k+p] \geq \mathbf{0}, \quad p = 0, \dots, P. \end{aligned} \tag{13}$$

Here, compared to the previous formulation in (10) where only the current projection residual at time t_k is optimized, the minimization is carried out over a temporal preview window of P steps according to (12). The penalty factor ν in (13) is chosen to be large in order to suppress the auxiliary control inputs $\tilde{\lambda}[k+p]$ since their role is to deviate the system only if additional constraints need to be obeyed as discussed below.

5.3 State Constraints for Obstacle Avoidance

Here, the goal is to avoid obstacles in state space. For simplicity, we only consider constraints on the positions $\boldsymbol{q} = [q_1, \dots, q_F]^T$ of the F state vectors $\boldsymbol{x}_f$ in (13) although velocity constraints on $\dot{\boldsymbol{q}} = [\dot{q}_1, \dots, \dot{q}_F]^T$ can be handled in the same fashion. To ensure convexity, we only consider linear state constraints of the form $\boldsymbol{h}^T \boldsymbol{q} + e \leq 0$ which facilitates the solution of the underlying optimization problem. Here, $\boldsymbol{h} \in \mathbb{R}^F$ is a unit normal vector and e is a scalar offset.

In our framework one DS in (11) is learned to guide each DoF. At this point, the state evolutions of these DS are independent, there is only a potential temporal coupling via shared phase variables s driving the inputs $\boldsymbol{u}[k]$ in (11). Here, we couple J systems in (11) via the state constraints which requires extending the MPC scheme in (13) as shown below

$$\underset{\boldsymbol{\Gamma}_1, \dots, \boldsymbol{\Gamma}_F}{\text{minimize}} \quad \sum_{f=1}^{F} \left(\|\Delta \boldsymbol{X}_f[k]\|_H^2 + \kappa \boldsymbol{l}_f[k]^T \boldsymbol{\lambda}[k] + \nu \sum_{p=0}^{P} \tilde{\lambda}_f[k+p]^2 \right) \tag{14}$$

subject to

$$\begin{aligned} & \mathbb{1}^T \boldsymbol{\lambda}_f[k+p] = 1, \quad p = 0, \dots, P, \quad f = 1, \dots, F \\ & \boldsymbol{\lambda}_f[k+p] \geq \mathbf{0}, \quad p = 0, \dots, P, \quad f = 1, \dots, F \\ & \boldsymbol{H}[k+p]^T \boldsymbol{q}[k+p] + \boldsymbol{E}[k+p] \leq 0, \quad p = 1, \dots, P. \end{aligned} \tag{15}$$

We consider C constraints at a given time step in the preview window, their normals are collected in the matrix $\boldsymbol{H}[k+p] = [\boldsymbol{h}_1[k+p], \dots, \boldsymbol{h}_C[k+p]]^T \in \mathbb{R}^{C \times F}$, $\boldsymbol{E}[k+p] = [e_1[k], \dots, e_C[k]]^T$ holds the corresponding offsets. To account for the coupling introduced by $\boldsymbol{q}[k+p]$ in (15), we have to consider the evolutions of each of the F states $\boldsymbol{x}_f[k+p]$

$$\begin{bmatrix} \boldsymbol{x}_f[k+1] \\ \boldsymbol{x}_f[k+2] \\ \vdots \\ \boldsymbol{x}_f[k+P] \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{A}}^1 \\ \bar{\boldsymbol{A}}^2 \\ \vdots \\ \bar{\boldsymbol{A}}^P \end{bmatrix} \boldsymbol{x}[k] + \boldsymbol{\Upsilon} \boldsymbol{\Gamma}_{\boldsymbol{f}},$$

where the matrix $\boldsymbol{\Upsilon} \in \mathbb{R}^{2P \times (D+1)(P+1)}$ is given as

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \bar{\boldsymbol{B}}\boldsymbol{\mu}[k] & \mathbf{0} & \dots & \dots & \mathbf{0} \\ \bar{\boldsymbol{A}}\bar{\boldsymbol{B}}\boldsymbol{\mu}[k] & \bar{\boldsymbol{B}}\boldsymbol{\mu}[k+1] & \mathbf{0} & \dots & \vdots \\ \vdots & \vdots & \vdots & \mathbf{0} & \vdots \\ \bar{\boldsymbol{A}}^{P-1}\bar{\boldsymbol{B}}\boldsymbol{\mu}[k] & \bar{\boldsymbol{A}}^{P-2}\bar{\boldsymbol{B}}\boldsymbol{\mu}[k+1] & \dots & \bar{\boldsymbol{B}}\boldsymbol{\mu}[k+P-1] & \mathbf{0} \end{bmatrix}.$$

Note that the state coupling is only introduced in the constraints of (14), not the objective function which is simply a sum over the objectives in (13). Thus, if no constraint in (15) is active at a given time step, the resulting behavior is identical to the one produced by the uncoupled scheme in (13) and resembles the learned trajectories. Only if constraints in (15) are active, the auxiliary controls $\tilde{\lambda}_f[k+p]$ cause deviations in order to satisfy these constraints. Considering the choice of objective function in (14) and assuming a long enough preview horizon, the stability of the proposed controller can be guaranteed [39]. In the tests reported in Section 6.3, we experimented with horizon lengths of $P = 5$ and $P = 10$ time steps which led to stable behavior.

A remaining issue is how to extract appropriate spatio-temporal constraints for obstacle and self-collision avoidance from the robot's environment. This is an open research question and is out of the scope of this work. Previous works suggest heuristics based on simplified pre-planned paths [33,34]. Here, we only consider a point-robot model and introduce a simple heuristics in order to be able to verify our MPC scheme in Section 6.3. We assume that an obstacle is represented as a convex hull in $\mathcal{H}$-representation, given as a set of bounding hyperplanes $\mathcal{H}_h = \{(\boldsymbol{h}_h, e_h)\},\ h = 1, \dots, H$, where $\boldsymbol{h}_h \in \mathbb{R}^F$ denote the associated unit normal vectors and e_f the corresponding distances to the origin.

At each time step t_k we want to determine whether to augment $(\boldsymbol{H}[k+P], \boldsymbol{E}[k+P])$ in the optimization problem in (14) with a new constraint at the end of the preview horizon (*i. e.*, at time t_{k+P}). To this end, we formulate the following Linear Program (LP)

$$\begin{aligned} &\underset{\xi \in \mathbb{R}}{\text{minimize}} \quad \xi \\ &\text{subject to} \\ &\boldsymbol{h}_h^T \left(\boldsymbol{q}[k+P] + \xi \dot{\boldsymbol{q}}[k+P] + e_h\right) \geq 0, \quad h = 1, \dots, H, \\ &\xi \geq 0, \end{aligned} \tag{16}$$

which projects the state $\boldsymbol{q}[k+P]$ along the ray corresponding to the velocity $\dot{\boldsymbol{q}}_{k+P}$ onto the obstacle as illustrated in Fig. 5. If the above LP is feasible (*i. e.*, the state evolution "heads towards" the obstacle), the hyperplane containing the projection forms a new constraint in (15).

The computational load of the presented MPC 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 J QP's according to (14) and an LP according to (16) is required.

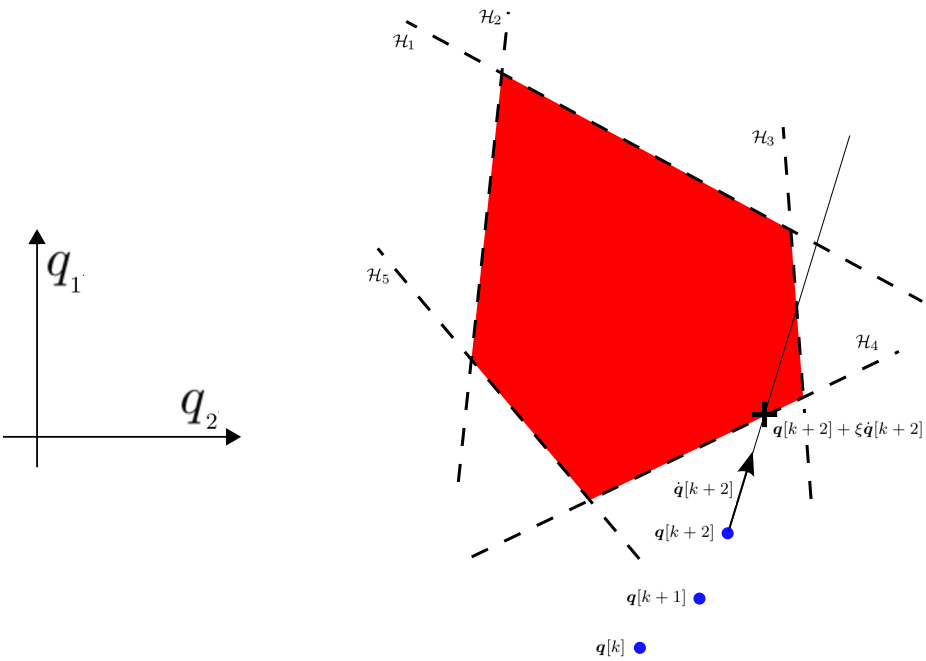
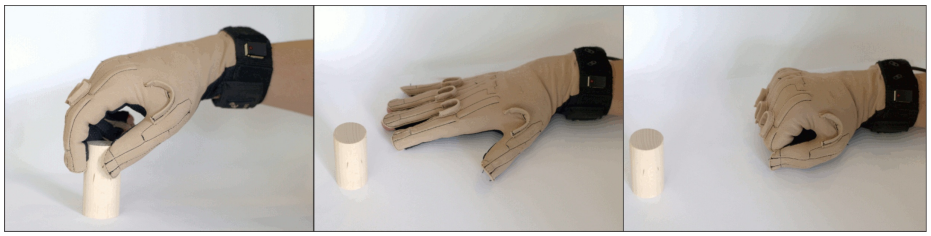


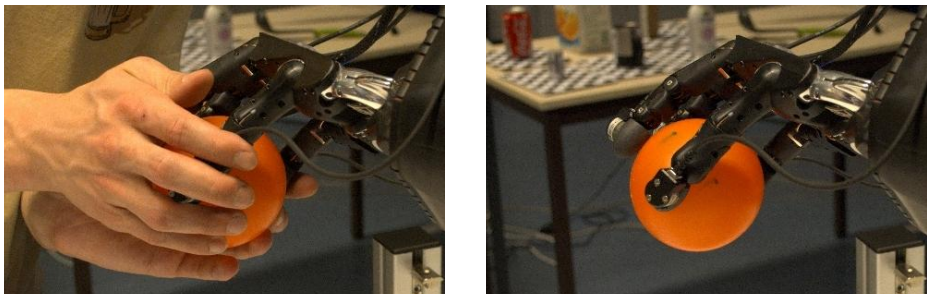
Fig. 5 *Finding Constraints for obstacle avoidance:* Shown is an example in a 2-dimensional configuration space, (*i.e.*, $F = 2$) with a preview window size of $P = 2$. The velocity ray at $\boldsymbol{q}[k+2]$ intersects hyperplane $\mathcal{H}_4$ at the point $\boldsymbol{q}[k+2] + \xi\dot{\boldsymbol{q}}[k+2]$, which is indicated by the black cross. Thus the constraint with normal $\boldsymbol{o}_4$ and offset f_4 associated with $\mathcal{H}_4$ is added to (15).

6 Evaluation

In this section we evaluate, by means of simulations and test runs on the Shadow Robot platform, the application of the suggested methods to offline learning of motion primitives from experimental data and the usage of these primitives for real-time motion control. To this end we used a sensorized glove to record taxonomic grasps on two cylindrical objects with different diameters as shown in Fig. 6(a). Opposed to [16], where we only performed one grasp type, in this work we chose to evaluate the approach on the following nine grasp types according to [9]: Tripod, Parallel Extension, Palmar Pinch, Large Diameter, Small Diameter, Lateral, Precision Sphere, Power Sphere and Inferior Pincer. 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 3, the trajectories were re-sampled with a number of $M = 100$



(a) *Teaching grasp motions*



(b) *Teaching final grasp configurations*

Fig. 6 *Teaching procedure:* (a) 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, grasps according to the taxonomy in [9] were performed. (b) Corresponding final grasp configurations were obtained by kinesthetic teaching and recording the robot's hand joint encoder values.

points each. A standard PC equipped with 6 GB memory and a 3.40 GHz Intel i7-2600 CPU was used to generate the presented results.

6.1 Reproduction and Generalization Capabilities

Here, the aim is to assess the introduced offline DMP learning scheme in (6). For the $F = 20$ DoF of the Shadow hand we used, for each of the aforementioned nine grasp types, demonstrated trajectories to estimate the free parameters of 20 motion primitives in (8) as described in Section 4.3. Thus, a total of 180 trajectories were used for the evaluation, the utilized fixed parameters are summarized in Table 1 in Appendix B. The constrained nonlinear least squares problems in (6) were solved with a Sequential Quadratic Programming (SQP) algorithm, utilizing the ACADO Toolkit [40].

In order to quantify the reproduction capabilities of the learned DMP, we reproduced the demonstrated trajectories by integrating (8) starting from the same initial values as the corresponding demonstrations. We experimented with different numbers N of basis functions in (3) and compared to results generated with DMP learned with fixed basis function parameters as in [12]. The resulting position- and velocity mean square errors (MSE), as well as the computation times for solving (6) for different numbers N of basis functions are summarized in Table 2 in Appendix B. Additionally, the position/velocity MSE are also depicted in Fig. 7. It is evident that, for small numbers of employed basis functions, the nonlinear

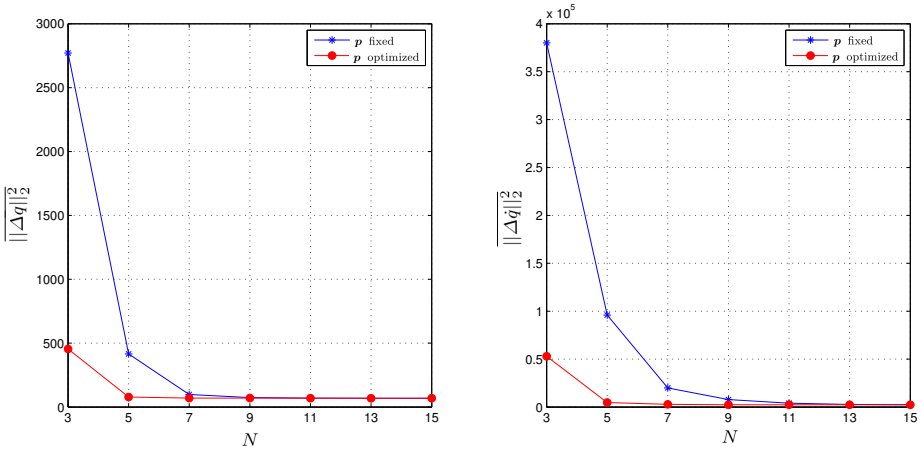
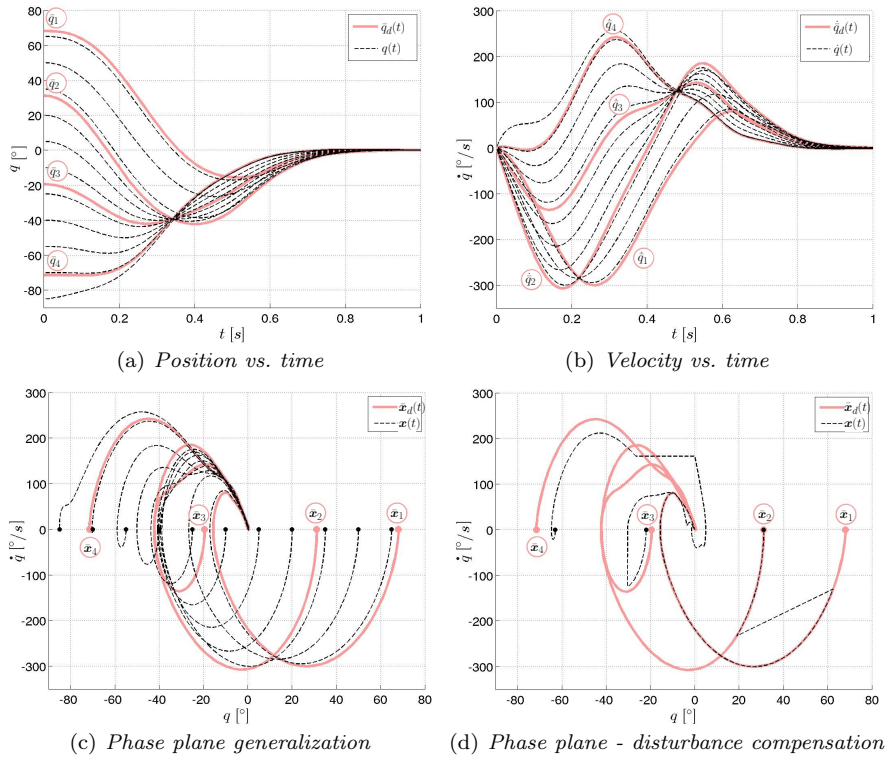


Fig. 7 *Reproduction quality:* Illustrated are the MSE describing the deviation from the trajectories produced by the systems in (8) from the learning data. Shown are the MSE for position (left) and velocity (right) for different numbers N of basis functions and for the basis function parameters $\boldsymbol{p}$ in (4) fixed/optimized.

learning scheme vastly outperforms linear learning with fixed basis function parameters. Also, the mean computation times for solving (6) while including the basis function parameters in the decision variables are within reasonable bounds (*e.g.* for $N = 7$ basis functions, the mean computation time is $6.2s$).

To gauge the generalization capabilities of the learned models for the considered point-to-point movements, we performed simulations by initializing our combined motion primitive formulation in (8) 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) during a tripod grasp are shown in Fig. 8. Depicted are the obtained position, velocity and phase plane curves. As argued in Section 5, 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 (10). States outside the convex hull over the references are attracted towards this convex hull according to dynamics governed by the matrix $\boldsymbol{A}$ in (8). 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 analyzed 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 (10) 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. 8(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



(a) *Position vs. time* (b) *Velocity vs. time*

(c) *Phase plane generalization* (d) *Phase plane - disturbance compensation*

Fig. 8 *Generalization over demonstrations and disturbance compensation:* Black dashed lines represent the trajectories obtained by simulating the dynamical system in (8), describing the tripod grasp 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. (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.

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 (8).

6.2 Verification on the Shadow Robot Hand/Arm Platform

Here, the goal is to demonstrate 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 and object perception 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. Compared to [16], where only

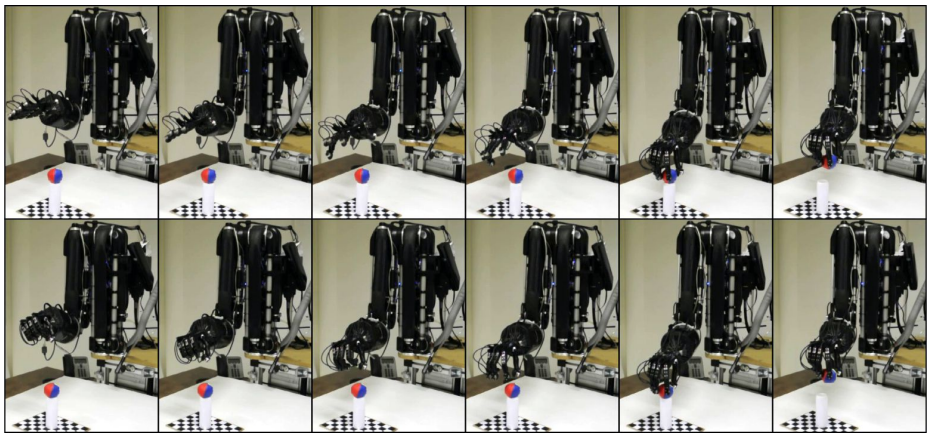
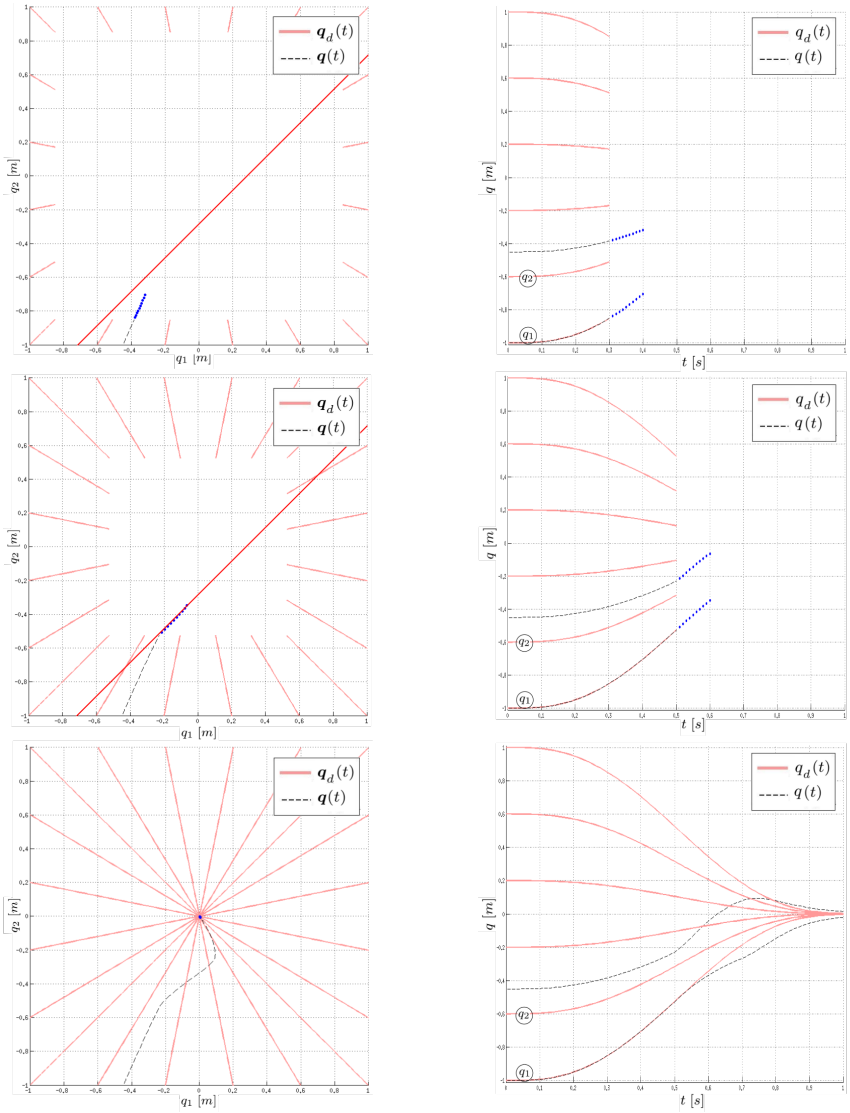


Fig. 9 *Tripod grasp primitives triggered from different initial configurations:* Synchronized finger joint movements are generated by means of integrating motion primitives corresponding to (8) which are driven by a common phase variable. Top row: Starting from an open hand configuration; bottom row: starting from a closed hand configuration.

a single grasp type was used, here we use the aforementioned nine grasp types considered feasible for the specific mechanical structure of the Shadow hand. 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. Desired final hand/arm configurations were obtained via kinesthetic teaching, as shown in Fig. 6(b), 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. 9. 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. Despite the obvious limitations in the low-level control as argued in [16], the grasping tasks were conducted successfully.

6.3 Obstacle Avoidance via State Constraints

Here, we want to discuss the behavior of the MPC scheme formalized in (14) under the influence of state constraints. To this, end we give two illustrating examples in a two-dimensional (*i.e.*, $F = 2$) obstacle space. The fixed parameters which were used in (14) are summarized in Table 3 in Appendix B. We use two primitives in (11), each of which learned from the same $D = 6$ synthetically generated examples of minimum-jerk trajectories. Figure 10 shows the evolution of the system at different points in time under influence of a single spatial constraint which is



(a) *Obstacle space* (b) *Position vs. time*

Fig. 10 *Constraint satisfaction:* Shown is the evolution of a 2D system driven by two primitives in (11) controlled by the MPC scheme in (14) with a preview window size of $P = 10$. The positions $\boldsymbol{q}$ computed by the controller are depicted with dashed black lines, the pink lines indicate the evolution of the encoded demonstrated position curves $\boldsymbol{q}_d$. The constraint vanishes after $t = 0.7s$ which allows the system to converge to its equilibrium. (a) shows the behavior in obstacle space, (b) depicts the according position curves.

active during part of the motion. The controller in (14) computes auxiliary control inputs $\tilde{\lambda}[k+p]$ such as to obey the constraint.

A second example is depicted in Figure 11. Here we employ the heuristic for the automatic extraction of appropriate constraint hyperplanes, as it was presented in

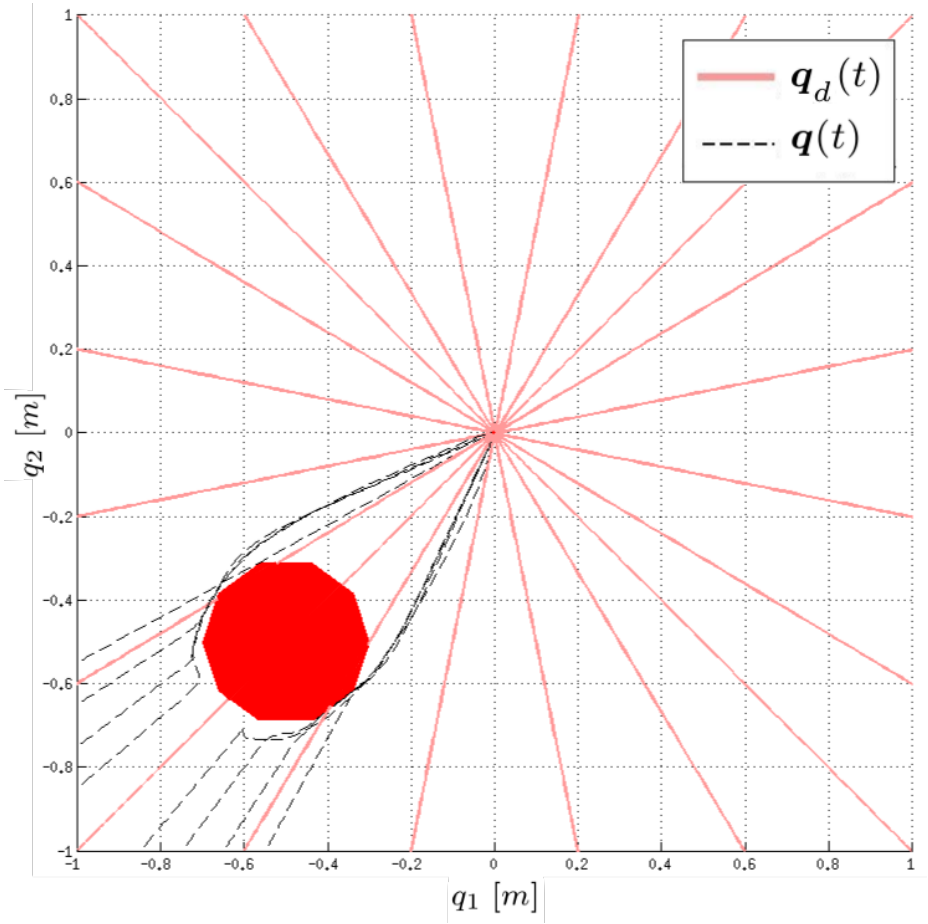


Fig. 11 *Automatic constraint update*: Shown is the obstacle avoidance behavior of the system controlled by (14) with a preview window size of $P = 5$, while using the heuristic according to (16) in order to extract constraints. The system was initialized with different start states, positions $\boldsymbol{q}$ computed by the controller are depicted with dashed black lines, the pink lines indicate the evolution of the encoded demonstrated position curves $\boldsymbol{q}_d$.

Section 5.3, to avoid a convex obstacle. Shown are the trajectories generated when starting from different points in the obstacle space.

7 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 introduce a MPC scheme based on 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, which allows to incorporate state constraints to account for additional tasks such as obstacle avoidance. The use of embedded optimization for addressing the on-line obstacle avoidance problem is a promising approach already heavily utilized in other scientific fields, this work is a first step towards a reactive on-line planning/control scheme. Future work will aim at extending the introduced obstacle avoidance scheme beyond the presently used point-robot model in order to make it applicable to real-life robot motion execution tasks.

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. 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 Intelligent Robots and Systems*, vol. 2, 2003, pp. 1444 – 1449.
2. 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.
3. K. Schittkowski, *Numerical Data Fitting in Dynamical Systems*. Kluwer Academic Publishers, 2002.
4. 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.
5. T. Flash and N. Hogans, "The coordination of arm movements: An experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, pp. 1688–1703, 1985.
6. R. Weitschat, S. Haddadin, F. Huber, and A. Albu-Schauffer, "Dynamic optimality in real-time: A learning framework for near-optimal robot motions," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 5636–5643.
7. S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.
8. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IJRR*, vol. 5, no. 1, pp. 90–98, 1986.
9. 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.
10. Shadow Robot Company, "The shadow dextrous hand." [Online]. Available: http://www.shadowrobot.com/hand/
11. A. Bernardino, M. Henriques, N. Hendrich, and J. Zhang, "Precision grasp synergies for dexterous robotic hands," in *Proc. of the IEEE International Conference on Robotics and Biomimetics*, 2013, pp. 62–67.
12. 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.

13. 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.
14. 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.
15. 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.
16. R. Krug and D. Dimitrov, "Representing movement primitives as implicit dynamical systems learned from multiple demonstrations," in *Proc. of the Int. Conf. on Advanced Robotics*, 2013, pp. 1–8.
17. 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.
18. P. Kormushev, S. Calinon, and D. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 3232–3237.
19. F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *11th IEEE-RAS Int. Conf. on Humanoid Robots*, 2011, pp. 231 – 238.
20. 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.
21. J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1 - 2, pp. 171 – 203, 2011.
22. B. Nemec, R. Vuga, and A. Ude, "Efficient sensorimotor learning from multiple demonstrations." *Advanced Robotics*, vol. 27, no. 13, pp. 1023–1031, 2013.
23. P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 763 – 768.
24. T. Matsubara, S. Hyon, and J. Morimoto, "Learning stylistic dynamic movement primitives from multiple demonstrations," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 1277–1283.
25. S. Calinon, Z. Li, T. Alizadeh, N. Tsagarakis, and D. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2012, pp. 323–329.
26. 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.
27. E. Gribovskaya, S. M. Khansari-Zadeh, and A. Billard, "Learning non-linear multivariate dynamics of motion in robotic manipulators," *IJRR*, vol. 30, no. 1, pp. 80 – 117, 2011.
28. 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.
29. A. Gams, B. Nemec, L. Zlajpah, M. Wachter, A. Ijspeert, T. Asfour, and A. Ude, "Modulation of motor primitives using force feedback: Interaction with the environment and bimanual tasks," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 5629–5635.
30. M. Saveriano and D. Lee, "Point cloud based dynamical system modulation for reactive avoidance of convex and concave obstacles," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 5380–5387.
31. S. V. Raković and D. Q. Mayne, "Robust model predictive control for obstacle avoidance: discrete time case," in *Assessment and Future Directions of Nonlinear Model Predictive Control*. Springer, 2007, pp. 617–627.
32. Z. Shiller, S. Sharma, I. Stern, and A. Stern, "Online obstacle avoidance at high speeds," *IJRR*, vol. 32, no. 9–10, pp. 1030–1047, 2013.
33. 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 Intelligent Robots and Systems*, 2012, pp. 5262– 5269.
34. 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.

35. H. V. S. Wangmanaopituk and W. Kongprawechnon, "Collaborative nonlinear model-predictive motion planning and control of mobile transport robots for a highly flexible production system," *Science Asia*, 2010.
36. M. Da Silva, Y. Abe, and J. Popović, "Simulation of human motion data using short-horizon model-predictive control," in *Computer Graphics Forum*, vol. 27, no. 2, 2008, pp. 371–380.
37. H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.
38. E. Rückert and A. d'Avella, "Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems," *Frontiers in computational neuroscience*, vol. 7, 2013.
39. D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
40. 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.

A Proof of Proposition 1

To prove Proposition 1 in Section 5 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 (8) are

$$\boldsymbol{x}_d[k+1] = \bar{\boldsymbol{A}}\boldsymbol{x}_d[k] + \bar{\boldsymbol{B}}u_d[k] \tag{17}$$

$$\boldsymbol{x}[k+1] = \bar{\boldsymbol{A}}\boldsymbol{x}[k] + \bar{\boldsymbol{B}}\boldsymbol{u}[k]^T\boldsymbol{\lambda}^*, \tag{18}$$

where $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are the respective state transition matrix and control matrix of the discrete system. Substituting (17) and (18) in (9) for time t_{k+1} results in

$$\Delta\boldsymbol{x}[k+1] = \bar{\boldsymbol{A}}\underbrace{(\boldsymbol{x}[k] - \boldsymbol{R}[k]\boldsymbol{\lambda}^*)}_{\Delta\boldsymbol{x}[k]}, \tag{19}$$

which confirms the first part of Proposition 1.

Furthermore, we note that if the projection residual $\Delta\boldsymbol{x}[k]$ in (19) is zero, the state $\boldsymbol{x}[k]$ can be expressed as a convex combination of the reference states in the columns of $\boldsymbol{R}[k]$. Thus, for $\Delta\boldsymbol{x}[k] = 0$, we can rewrite (18) as

$$\begin{aligned}\boldsymbol{x}[k+1] &= \bar{\boldsymbol{A}}\underbrace{\boldsymbol{R}[k]\boldsymbol{\lambda}^*}_{\boldsymbol{x}[k]} + \bar{\boldsymbol{B}}\boldsymbol{u}[k]^T\boldsymbol{\lambda}^* \\ &= \boldsymbol{R}[k+1]\boldsymbol{\lambda}^*\end{aligned}$$

which concludes the proof.

B Tables

Table 1 Fixed parameters used for the evaluation presented in Section 6.1

N	α	β	ϵ	Δc	$\boldsymbol{H}$	κ
5	-132.5	-23	10^{-4}	0.05	diag(100, 1)	0

Table 2 Demonstration reproduction results: Δq and $\Delta \dot{q}$ denote the position/velocity errors between the demonstrated data and the trajectories reproduced by the locally optimal DMP combination in (8), T_c denotes the computation time needed to solve (6). Superscripts $(\cdot)^F$ and $(\cdot)^O$ denote whether the results were generated with fixed or optimized basis function parameters.

N	$\|\|\Delta q^F\|\|_2^2$	$\|\|\Delta q^O\|\|_2^2$	$\|\|\Delta \dot{q}^F\|\|_2^2$	$\|\|\Delta \dot{q}^O\|\|_2^2$	T_c^F [s]	T_c^O [s]
3	2770.6±6802.7	454.7±1552.1	$3.8{\cdot}10^5{\pm}9.0{\cdot}10^5$	$5.3{\cdot}10^4{\pm}1.6{\cdot}10^5$	0.005±0.026	0.5±0.4
5	415.4 ±1094.6	78.1 ±170.5	$9.6{\cdot}10^4{\pm}2.5{\cdot}10^5$	$4694.3{\pm}1.1{\cdot}10^4$	0.006±0.002	2.1±4.0
7	98.4 ±217.4	70.1 ±150.7	$2.0{\cdot}10^4{\pm}4.8{\cdot}10^4$	2772.6±5470.8	0.006±0.002	6.2±9.4
9	74.4 ±160.8	69.7 ±150.5	$7841.9{\pm}1.8{\cdot}10^4$	2399.9±4931.8	0.006±0.002	30.9±63.9
11	70.4 ±152.4	69.3 ±149.4	4089.4±9328.8	2358.6±4798.7	0.006±0.002	58.8±92.13
13	69.1 ±149.9	68.8 ±149.1	2751.2±6087.1	2355.0±4927.6	0.007±0.004	26.9±50.0
15	68.8 ±149.3	68.9 ±149.1	2411.8±5184.8	2350.8±4786.2	0.007±0.003	62.2±79.4

Table 3 Fixed parameters used for the examples in Section 6.3

N	α	β	ϵ	Δc	$\boldsymbol{H}$	κ	ν	D
5	-132.5	-23	10^{-4}	0.05	diag(1, 1)	10^{-6}	10^4	6