Learning Quadrotor Maneuvers From Optimal Control and Generalizing in Real-Time

Teodor Tomić*, Moritz Maier* and Sami Haddadin

***Abstract*— In this paper, we present a method for learning and online generalization of maneuvers for quadrotor-type vehicles. The maneuvers are formulated as optimal control problems, which are solved using a general purpose optimal control solver. The solutions are then encoded and generalized with Dynamic Movement Primitives (DMPs). This allows for real-time generalization to new goals and in-flight modifications. An effective method for joining the generalized trajectories is implemented. We present the necessary theoretical background and error analysis of the generalization. The effectiveness of the proposed method is showcased using planar point-to-point and perching maneuvers in simulation and experiment.**

I. Introduction

Quadrotors belong to the group of unmanned aerial vehicles (UAVs) and vertical take-off and landing aircrafts (VTOLs). Due to their simple mechanical design, their broad availability, and mainly, their ability to hover, quadrotors are well suited for inspection, surveillance and aerial photography applications. Current research is focused on autonomous exploration of unknown environments in search and rescue scenarios, particularly without external position information, e.g. in GPS-denied areas [1], [2]. The generation of aerobatic, aggressive, or time-optimal maneuvers for quadrotors has so far been addressed by many researchers. The available approaches differ mainly in the description of the maneuver, online-offline capabilities, and optimality criteria.

A. Related work

Multiple approaches exist in literature to obtain trajectories for a quadrotor-like vehicle. In this work, we want the trajectory to drive the vehicle through predefined states, while maintaining input and state constraints.

The method described in [3] uses the flatness property of the quadrotor system to obtain a piecewise polynomial trajectory in the flat output (Cartesian position and yaw angle). Trajectories are constrained through a sequence of keyframes, and are used to obtain feed-forward control inputs. The trajectory snap, which corresponds to angular velocity, is minimized using offline linear programming. Input constraints are handled by temporal scaling of the trajectory. The control input is assumed to be polynomial, and its degree depends on the number of constraints in the keyframes. Spatial scaling of the whole trajectory is possible in-flight.

In [4], the authors define a maneuver as a sequence of discrete motions. An outdoor backflip maneuver is divided into three stages (impulse, drift, recovery). The safety and attainability of the desired sets in state space are ensured by using the backwards reachability concept. The trajectories are highly parameter dependent. Therefore, the whole set must be recalculated if a parameter change occurs. The generated maneuvers are not optimal in any sense.

Real-time generation of time-optimal point-to-point quadrotor trajectories has been investigated in the literature [5], [6], [7]. Therein, a closed-form solution minimizing the maximum acceleration has been found. The effectiveness of this approach was shown shown in terms of real-time interception maneuvers [6] and coordinated ball throwing and catching [7]. An indirect optimal control method is applied to the minimum time problem in [8].

It can be concluded that several effective methods exist for trajectory generation of isolated maneuvers (backflip, flip, point-to-point) and performance measures (minimum time). This has been achieved primarily by using simplified planar models. However, handling arbitrary state and input constraints is still limited. Hence, the real-time generation of optimal trajectories for arbitrary maneuvers under general state and input constraints is still an unsolved problem.

B. Scope and contribution of this work

We aim to solve the problem of arbitrary performance measures and constraints by solving an optimal control problem offline for a grid of goals. The results are then learned using a machine learning technique. The learned trajectories implicitly include the constraints used in the optimal control problem. This allows for online generalization of the optimal results to obtain near-optimal trajectories. We base our work on a similar approach used for robotic arms [9].

The paper is structured as follows. We formulate a maneuver as a general optimal control problem in Section II. It can therefore consist of multiple phases in order to include, e.g. *via*-points, as well as arbitrary state and input constraints. The solution is obtained offline for a set of trajectories, using a general-purpose optimal control solver. Second, the obtained trajectories are learned using Dynamic Movement Primitives (DMPs) [10] (Section III). In this representation, the trajectories can be generalized online to new goals. We developed in-flight DMP modifications to include joining of trajectories. The effectiveness of the approach is shown for different simulations and experiments in Section IV.

* The authors contributed equally to this paper
{teodor.tomic,moritz.maier}@dlr.de
German Aerospace Center (DLR) Robotics and Mechatronics Center (RMC), Münchener Straße 20, 82234 Weßling, Germany
sami.haddadin@irt.uni-hannover.de
Institute of Automatic Control, Leibniz University Hanover (LUH), Appelstr. 11, 30167 Hanover, Germany

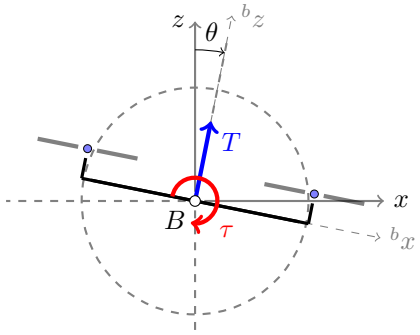


Fig. 1. Simplified model of the quadrotor motion in the (x, z)-plane. The body frame is defined by the axes ${}^b x$-${}^b y$-${}^b z$.

II. Modeling and Optimal Control

A. Planar quadrotor model

In this paper, we use a planar quadrotor model, see Fig. 1. This is a well-established approach which can be found in [4], [11], [12], [13]. The attitude of the quadrotor is described by the angle θ about the body axis ${}^b y$, x and z are the position of the center of gravity in the inertial (x, z)-plane, $\dot{\theta}$ is the angular velocity about the ${}^b y$-axis. The first control input is the collective thrust divided by the quadrotor's mass $u_1 = \frac{T}{m}$. The second control input is $u_2 = \frac{\tau}{J}$, where τ and J are the torque and inertia about the body y-axis, respectively. Using the state $\boldsymbol{x} = [x\, \dot{x}\, z\, \dot{z}\, \theta\, \dot{\theta}]^T$ and control $\boldsymbol{u} = [u_1\, u_2]^T$, the equations of motion for the planar model are

$$\begin{aligned} \ddot{x} &= u_1 \sin(\theta) \\ \ddot{z} &= u_1 \cos(\theta) - g \\ \ddot{\theta} &= u_2. \end{aligned} \tag{1}$$

It has previously been shown that the quadrotor system is differentially flat [14]. This means that the control inputs $\boldsymbol{u}$ can be algebraically computed from desired accelerations. Furthermore, $\boldsymbol{u}$, θ, $\dot{\theta}$ and $\ddot{\theta}$ can be calculated from given desired accelerations $\ddot{x}_d, \ddot{z}_d$. By algebraic manipulation we obtain

$$u_1 = \sqrt{(\ddot{z}_d + g)^2 + \ddot{x}_d^2}, \tag{2}$$

$$\theta = \operatorname{atan2}\left(-\ddot{x}_d, g + \ddot{z}_d\right), \tag{3}$$

where atan2 is the four-quadrant nonsingular variant of the arctangens function. The angular velocity $\omega = \dot{\theta}$ can be derived by differentiating (3), for which the jerk is needed. Computing $u_2 = \ddot{\theta}$ requires the second derivative of the acceleration (snap) of the trajectory. This shows that only the flat output and its derivatives are needed for successful reproduction of a maneuver. Next we formulate optimal control problems to compute the maneuvers using the presented model.

B. Optimal control problem

In the first step of our approach, we solve a set of optimal control problems for a maneuver using model (1).

Definition 1 (Optimal control problem): Find a control $\boldsymbol{u}^*$ which causes the system

$$\dot{\boldsymbol{x}} = \boldsymbol{a}(\boldsymbol{x}(t), \boldsymbol{u}(t)),$$

with $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, $t \in [t_0, t_f]$ subject to the inequality path constraints

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max}, \quad \boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max},$$

and the boundary conditions $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, $\boldsymbol{x}(t_f) = \boldsymbol{x}_f$, to follow a trajectory $\boldsymbol{x}^*$ that minimizes the optimality criterion

$$\mathcal{J}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \mathcal{J}_M\left(\boldsymbol{x}(t_f), t_f\right) + \int_{t_0}^{t_f} \mathcal{J}_L\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) \mathrm{d}t.$$

Model (1) can be rewritten as

$$\dot{\boldsymbol{x}} = \boldsymbol{a}(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} \dot{x} \\ u_1 \sin(\theta) \\ \dot{z} \\ u_1 \cos(\theta) - g \\ \dot{\theta} \\ u_2 \end{pmatrix}, \tag{4}$$

with constraints on the controls $\boldsymbol{u} = [u_1\, u_2]^T$ defined as

$$\frac{\overline{T}}{m} \leq u_1 \leq \frac{\underline{T}}{m}, \qquad |u_2| \leq \frac{\overline{\tau}}{J}, \tag{5}$$

where $\overline{T}$ and $\underline{T}$ are the maximum and minimum thrust, respectively. $\overline{\tau}$ is the maximum absolute torque. Table I shows the system parameters and constraints used throughout the paper.

For the purposes of learning meaneuvers, we treat the solution of the optimal control problem as a black box. How the solution is obtained is not critical for the methods presented below. In this paper, we solve the optimal control problem numerically. We use the Matlab package GPOPS, which implements the Gauss pseudospectral method [15], [16], [17]. We are principally interested in the minimum time problem, hence $\mathcal{J}_M = t_f$ and $\mathcal{J}_L = 0$. However, the pseudospectral optimal solution may oscillate and overshoot for bang-bang type controls due to the Gibbs phenomenon. We therefore include a regularization term in the goal function to obtain smoother controls. This is a consequence of using a pseudospectral solver. If a different solver is used, the regularization term may not be needed. Our final goal function is therefore chosen as

$$\mathcal{J}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = t_f + \int_0^{t_f} \left(r_1 u_1^2(t) + r_2 u_2^2(t)\right) \mathrm{d}t, \tag{6}$$

where r_1 and r_2 are small regularization parameters. These are chosen experimentally to minimize the effect of Gibbs' phenomenon, i.e. obtain smooth inputs. Using a general-purpose optimal control solver allows the definition of arbitrary maneuvers with state and input constraints. GPOPS can e.g. also solve multi-phase problems. This allows each maneuver to contain any feasible states during the maneuver, e.g. both a velocity and pitch angle can be defined at a phase boundary.

C. Maneuvers

Although the approach presented in this paper is valid for general maneuvers, we investigated three kinds of maneuvers so far: point-to-point, perching, and flip. A maneuver may contain multiple phases. In that case, a *via* condition is imposed, specifying the state at the respective phase bound-

TABLE I

SYSTEM PARAMETERS, STATE AND INPUT CONSTRAINTS USED FOR THE OPTIMAL CONTROL PROBLEM AND THE PLANAR QUADROTOR MODEL.

System parameters	State constraints	Input constraints
$m = 0.5\,\text{kg}$	$\bar{\dot{x}} = \bar{\dot{z}} = 10\,\text{m/s}$	$\overline{T} = 12\,\text{N}$
$J = 3 \times 10^{-3}\,\text{kgm}^2$	$\underline{\dot{x}} = \underline{\dot{z}} = -10\,\text{m/s}$	$\underline{T} = 1\,\text{N}$
$g = 9.81\,\text{m/s}^2$	$\bar{\dot{\theta}} = 300\,°/\text{s}$	$\overline{\tau} = 0.2\,\text{Nm}$
	$\underline{\dot{\theta}} = -300\,°/\text{s}$	$\underline{\tau} = -0.2\,\text{Nm}$

TABLE II

OVERVIEW OF MANEUVERS CONSIDERED IN THIS PAPER

Maneuver	Via condition	End condition
Point-to-point	–	$\boldsymbol{p}_f = [x_g\, z_g\, 0]^T$ $\dot{\boldsymbol{p}}_f = [0\,0\,0]^T$
Perching	–	$\boldsymbol{p}_f = [x_g\, z_g\, \frac{\pi}{2}]^T$ $\dot{\boldsymbol{p}}_f = [0\,0\,0]^T$
Flip	$\boldsymbol{p}_{\text{via}} = [x_{\text{via}}\, z_{\text{via}}\, \pi]^T$	$\boldsymbol{p}_f = [x_g\, 0\, 2\pi]^T$ $\dot{\boldsymbol{p}}_f = [0\,0\,0]^T$

ary. The conditions are chosen intuitively by the maneuver designer.

For notational simplicity we will define the maneuver constraints through the position $\boldsymbol{p} = [x\, z\, \theta]^T$ and velocity $\dot{\boldsymbol{p}} = [\dot{x}\, \dot{z}\, \dot{\theta}]^T$. All maneuvers start at initial conditions $\boldsymbol{p}_0 = [0\,0\,0]^T$, $\dot{\boldsymbol{p}}_0 = [0\,0\,0]^T$, with the via and end conditions specified in Table II. Fig. 2 shows the trajectories of three maneuvers considered in this paper.

a) Point-to-point maneuver: The goal is to reach a desired location $\boldsymbol{p}_f = [x_g\, z_g\, 0]^T$ in minimum time. Zero velocity is imposed at start and end of the maneuver.

b) Perching maneuver: A perching maneuver is a special case of a point-to-point flight with a nonzero final attitude angle, for instance $\theta_g = \pm\frac{\pi}{2}$. It illustrates that it is possible to define fixed states of the quadrotor at specified instants of time. If a perching mechanism is available, the quadrotor would hold to a surface at the end of the maneuver. Otherwise, stabilization to hover must be done.

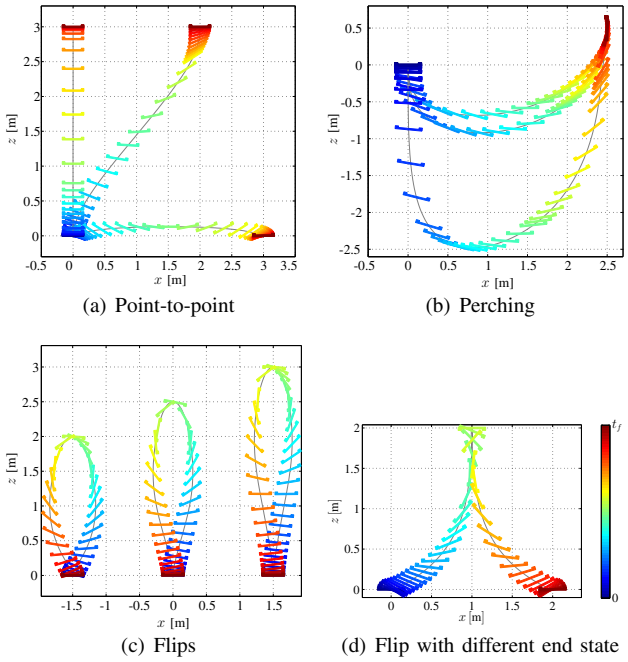
c) Flip: During the maneuver, the quadrotor passes a specific *via* point with the attitude of $\theta_{\text{via}} = \pi$ (upside down). In order to make a full rotation, we set the end pitch angle to be $\theta_f = 2\pi$, with zero velocities to come to hover.

For each maneuver, we solve the optimal control problem for a series of goal points $\boldsymbol{p}_f$. We obtain a set of optimal trajectories, which are learned using a representation that allows generalization to new goals.

III. MANEUVER LEARNING AND GENERALIZATION

A. Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) were first introduced in [10]. A DMP consists of a stable possibly nonlinear dynamic attractor system, which is perturbed by a learned external force such that the system performs a desired movement. Due to constant inertia of the quadrotor, the DMP system is linear in our case. The external force is represented as a Gaussian basis, enabling to learn and



(a) Point-to-point (b) Perching

(c) Flips (d) Flip with different end state

Fig. 2. Position trajectories of the maneuvers considered in this paper, obtained by a general-purpose optimal control solver. The color of the quadrotor contour corresponds to the defined instants of time at which a quadrotor is plotted. Fig. 2(a) shows point-to-point trajectories from $\boldsymbol{p}_0 = [0\,0\,0]^T$ to different goal points. Fig. 2(b) depicts perching trajectories for different maximum angular velocities $\bar{\dot{\theta}}$. The values differ from the one specified in Table I. Fig. 2(c) shows flip trajectories for different flip heights. Fig. 2(d) depicts a flip trajectory with a *via* point and different goal point.

approximate it from training data. Training trajectories can, for example, be obtained from a recorded movement, which is a common approach in the field of imitation learning [10], [18], [19], from optimal control solutions [20] or any other type of trajectory generator. DMPs are applied to a variety of robotics problems [21], [22], [23], since they can produce both discrete or rythmic movements that can quickly be adapted to changes in a dynamic environment [24]. Different motion primitives have been applied to quadrotor helicopters in [11], [25], [26].

We utilize the critically damped second-order mass-spring-damper formulation

$$m\ddot{\boldsymbol{r}} + d\dot{\boldsymbol{r}} + \kappa(\boldsymbol{r} - \boldsymbol{r}_g) + \boldsymbol{f}_t = \boldsymbol{0}, \tag{7}$$

where $\boldsymbol{r} = [x\, y\, z]^T$ is the trajectory in the flat outputs, m is the quadrotor mass, $\kappa > 0$ is a stiffness parameter, $d = 2\sqrt{m\kappa}$ is the critical damping parameter, and $\boldsymbol{f}_t = [f_{t,x}\, f_{t,y}\, f_{t,z}]^T$ is the trajectory force. We use all three spatial coordinates, since the planar maneuvers can be executed in an arbitrary vertical plane using a proper transformation of the force $\boldsymbol{f}_t$. For six degrees of freedom maneuvers, (7) can be extended by the yaw angle. By formulating the trajectory in this way, we can encode position $\boldsymbol{r}$, velocity $\dot{\boldsymbol{r}}$, and acceleration $\ddot{\boldsymbol{r}}$ using only the learned force $\boldsymbol{f}_t$. The goal position $\boldsymbol{r}_g$ allows the generalization of the movement primitive to new goals. For $\boldsymbol{f} = \boldsymbol{0}$, it can be shown that $\boldsymbol{r}_g$ is the asymptotically BIBO stable equilibrium.

To learn and approximate the perturbation force $\boldsymbol{f}_t$ which

is needed to produce a desired trajectory $\boldsymbol{r}(t)$ using the mass-spring-damper system (7), $\boldsymbol{f}_t$ is encoded into a normalized Radial Basis Function (RBF) with N basis functions [10] that is defined as

$$\boldsymbol{f}_{\approx}(s) = \frac{\sum_{i=1}^{N} w_i \Psi_i(s)}{\sum_{i=1}^{N} \Psi_i(s)} s, \tag{8}$$

where the Gaussian basis is defined as

$$\Psi_i(s) = \exp\left(-h_i(s-c_i)^2\right). \tag{9}$$

Its center points are located at

$$c_i = \exp\left(-\alpha_s \frac{i-1}{N-1}\right). \tag{10}$$

The widths are defined as

$$h_i = \begin{cases} \frac{\beta_s}{(c_{i+1}-c_i)^2} + \gamma_s, & i = 1, ..., N-1, \\ h_{N-1}, & i = N. \end{cases} \tag{11}$$

The parameters $\beta_s > 0$ and γ_s are used to adjust the shape of the basis functions, and $\alpha_s > 0$ determines the locations of their centers. The path parameter s is a normalized time coordinate. It is defined by the differential equation [18]

$$\dot{s} = -\frac{\alpha_s}{\lambda_t} s, \quad s(0) = 1, \quad s(t_f) = 0. \tag{12}$$

Definition of the DMP through s enables temporal scaling of the trajectories. Furthermore, as s decreases towards zero, the perturbation force $\boldsymbol{f}_{\approx}$ also decreases towards zero. Then, (7) is still a stable attractor. Here, $\lambda_t > 0$ is an additional temporal scaling factor, which should not be mistaken with the duration t_f of a movement. Within the learning process we set $\lambda_t = 1$. For reproduction of learned trajectories it is used to apply temporal scaling. Note that (10) and (11) represent one possible choice for the centers and widths, which results for $\beta_s \neq 0$ and the path parameter (12) in identical and equally distributed Gaussian basis functions with respect to time t [20].

B. Learning and optimization

The basis function approximation of the target force is linearly dependent on the weights. Therefore, it can be written as

$$\boldsymbol{A}\boldsymbol{w} = \boldsymbol{f}_{\approx} \approx \boldsymbol{f}_t, \tag{13}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \frac{\Psi_1(s_1)}{\sum_{i=1}^{N}\Psi_i(s_1)} s_1 & \dots & \frac{\Psi_N(s_1)}{\sum_{i=1}^{N}\Psi_i(s_1)} s_1 \\ \dots & \dots & \dots \\ \frac{\Psi_1(s_M)}{\sum_{i=1}^{N}\Psi_i(s_M)} s_M & \dots & \frac{\Psi_N(s_M)}{\sum_{i=1}^{N}\Psi_i(s_M)} s_M \end{bmatrix}. \tag{14}$$

Here, s_j denotes the value of the path parameter at $t = j\Delta t$, with Δt being the trajectory sampling step, and M the number of samples. The least-squares goal function

$$\Gamma = \sum_{i=1}^{M} (\boldsymbol{f}_{t,i} - \boldsymbol{f}_{\approx,i})^2$$

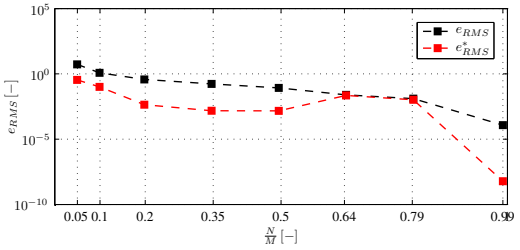


Fig. 3. Root mean square fitting error of the DMP force $\boldsymbol{f}_t$ using different number of basis functions with (e^*_{RMS}) and without (e_{RMS}) optimized parameters α_s, β_s, γ_s. The parameters are optimized on a family of R trajectories. Parameter optimization can improve the fitting by several orders of magnitude.

can thus be optimized by the left Moore-Penrose pseudo-inverse, i.e. linear regression

$$\boldsymbol{w}^* = \boldsymbol{A}^+ \boldsymbol{f}_t. \tag{15}$$

The quality of the approximation depends significantly on α_s, β_s, γ_s, $\hat{\kappa}$, and N. In practice, $\hat{\kappa}$ and N have to be adjusted manually such that the target force f_t is not oscillating. Additionally, to reproduce the original input data with higher accuracy, an optimization [27] of the three tuning parameters α_s, β_s, γ_s, that minimizes the goal function Γ, was done using the Matlab function *fmincon*. Since this provides only local convergence, a good initial parameter estimate is needed. The effect of using optimized parameters on the fitting accuracy is shown in Fig. 3. Fig. 4 illustrates the learning scheme for fitting one training trajectory.

C. Reproduction of trajectories

Having obtained the optimal weights w_i^* of the Gaussian basis, the learned trajectory can be reproduced by numerically integrating

$$m\ddot{\boldsymbol{r}} = -\hat{d}(t)\dot{\boldsymbol{r}} - \hat{\kappa}(t)(\boldsymbol{r} - \boldsymbol{r}_g) - \boldsymbol{f}_{\approx}, \tag{16}$$

with initial conditions $\dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0$, $\boldsymbol{r}(0) = \boldsymbol{r}_0$. Here, $\hat{d}(t)$ and $\hat{\kappa}(t)$ are time-varying damping and stiffness defined below. We use the approximated perturbation force $\boldsymbol{f}_{\approx}$ as an input. For reproduction, we note that the trajectory will reach the goal also for nonzero initial conditions $\boldsymbol{r}_0$ and $\dot{\boldsymbol{r}}_0$. This feature is heavily used in imitation learning [10], [18], [19]. However, we avoid this approach since the learned optimal solutions are only valid for the respective initial conditions. We extend the DMP approach presented above with a time-varying stiffness $\hat{\kappa}(t)$, which is continuously increasing and bounded, instead of being constant [20]. This attenuates the problem of the spring term $\kappa(\boldsymbol{x}(t=0) - \boldsymbol{x}_g)$ in (7), which produces a jump in the acceleration at the start of the trajectory. We use the stiffness

$$\hat{\kappa}(t) = \kappa\left(1 - \exp\left(-\frac{t}{k\, t_f}\right)\right), \tag{17}$$

wherein k is a positive constant tuning parameter, κ is the upper bound of the resulting stiffness and t_f is the duration of the trajectory. The damping $\hat{d}(t) = 2\sqrt{m\hat{\kappa}}$ then also becomes time-varying to maintain critical damping along the trajectory. For a maneuver that exceeds t_f, $\hat{\kappa}(t)$ converges

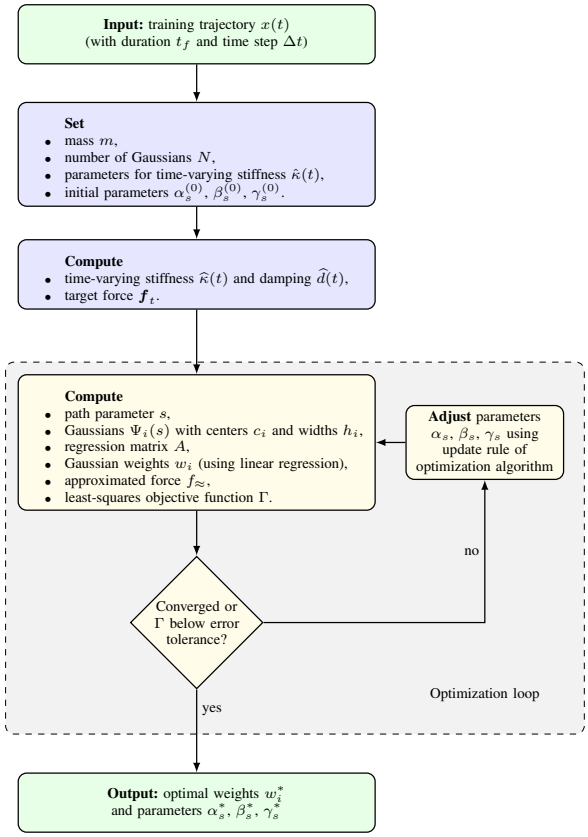


Fig. 4. Flowchart of the complete learning scheme including an optimization loop for the Gaussian basis functions.

to the upper bound κ. The parameter k can be obtained by evaluating (17) for $\kappa(t) = \kappa_d(t_d)$, i.e. setting a desired stiffness κ_d at a specified time t_d.

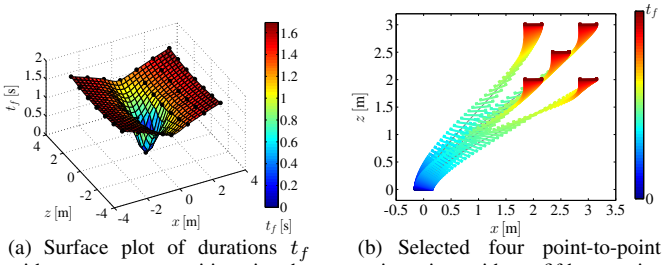
Furthermore, DMPs allow spatial scaling by a factor λ_s and temporal scaling by a factor λ_t. The scaled trajectory $\tilde{\boldsymbol{r}}$ will then be

$$m\lambda_t^2\lambda_s\ddot{\tilde{\boldsymbol{r}}}(\lambda_t t) = -\hat{d}\lambda_t\lambda_s\dot{\tilde{\boldsymbol{r}}} - \hat{\kappa}(\lambda_s\tilde{\boldsymbol{r}} - \boldsymbol{r}_g) - \boldsymbol{f}_{\approx}. \quad (18)$$

The trajectory is reproduced by numerically integrating (18) up to time $\lambda_t t_f$. For consistency, the temporal scaling factor λ_t has to be equal for the dynamic system (18) and for the path parameter s in (12).

D. Generalization to new goals

Using the DMP approach, it is possible to generalize the learned trajectories to new goals. By storing optimal control solutions for a spatial grid of goal points, we can generalize the DMP force for a new goal: for each point on the grid, we store the corresponding weights $\boldsymbol{w}$ and trajectory duration t_f (see Fig. 5(a)). Before fitting, all trajectories are scaled to a common duration t_0 with $\lambda_t = t_0/t_f$. For any new goal not on the grid, we use bilinear interpolation of the weights and the duration. Thereby we obtain the force $\boldsymbol{f}_{\approx}$ and time scaling parameter for the new goal. Hence, to adjust the trajectory to a new goal $\boldsymbol{r}_g$, the weights used in $\boldsymbol{f}_{\approx}$ are interpolated from existing approximations. Each trajectory in the grid is represented by a DMP. In effect, we interpolate



(a) Surface plot of durations t_f with respect to position in the (x, z)-plane obtained from training data using bicubic interpolation.

(b) Selected four point-to-point trajectories with a fifth one in-between.

Fig. 5. Interpolation of the weights from a mesh of point-to-point training trajectories. Only trajectories for $x \geq 0$ were computed, since the solutions are symmetric with respect to the z-axis. Fig. 5(a) shows a three-dimensional plot of the durations t_f with respect to the x- and z-positions of the trajectory goals, motivating the bilinear interpolation of the durations. The four trajectories in Fig. 5(b) were selected to show the bilinear interpolation used to generate the trajectory to the point in-between.

a new DMP for the new goal. This allows to cover a larger range of trajectories than with a simple DMP generalization approach. This requires that all solutions on the grid have the same number of basis functions. Therefore, we run batch fitting and parameter optimization for all trajectories on the grid. In this way, the obtained parameters will be optimal for the entire set of learned trajectories. The interpolated trajectory for a point inside a grid is shown in Fig. 5(b). This approach assumes linear behavior of the trajectory duration between the grid points. The assumption is a valid if the trajectories satisfy smoothness properties between the grid points.

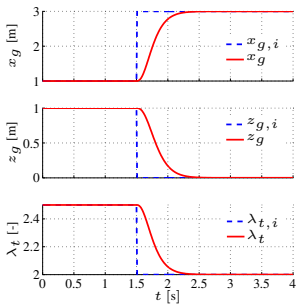
E. Joining of Trajectories

In order to generate a sequence of maneuvers, the trajectories produced by DMPs can be joined. Use cases are for instance a flight through via points, a double flip, or switching from the current trajectory to a different one. The latter implies that the goal point of the first trajectory is not reached yet. In the simplest case, the trajectories can be sequentially reproduced. This approach is referred to as simple joining [28], which leads to zero velocity at the transition point. A jump in acceleration also occurs due to the abruptly altered goal. This problem can be overcome by reformulating the DMP as a third-order system [28], [29]. Two DMPs can then be smoothly joined by overlapping the Gaussian basis kernels [28] or by explicitly calculating the initial conditions of the second DMP [29]. Alternatively, the second-order DMP weights can be adapted online [29].

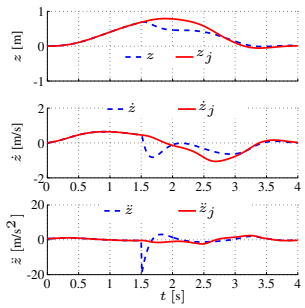
To avoid acceleration discontinuities when joining, we use an attractor to blend the goal positions at the transition point. The novel approach is inspired by proxy-based control [30], [31]. There, a virtual proxy object is attached to the controlled object in order to smoothly recover from large position errors. For joining, we blend the goal $\boldsymbol{r}_g$ from $\boldsymbol{r}_{g,1}$ to $\boldsymbol{r}_{g,2}$ using the system

$$m_p\ddot{\boldsymbol{r}}_g + \hat{d}_p\dot{\boldsymbol{r}}_g + \hat{\kappa}_p(\boldsymbol{r}_g - \boldsymbol{r}_{g,2}) = \boldsymbol{0}, \quad (19)$$

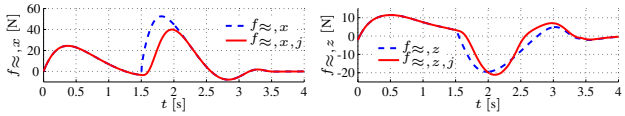
where m_p is the proxy mass, $\hat{\kappa}_p$ is the time-varying proxy stiffness and $\hat{d}_p$ is the time-varying proxy damping. Con-



(a) Blending of the goal positions x_g (top), z_g (middle) and the temporal scaling factor λ_t (bottom).



(b) Resulting joined trajectory in z-direction (top) with velocities $\dot{z}$ (middle) and accelerations $\ddot{z}$ (bottom).



(c) Blending of the external perturbation forces $f_{\approx,x}$ (left) and $f_{\approx,z}$ (right).

Fig. 6. Joining of two trajectories given in relative coordinates from $(0,0)^T$ to $(1,1)^T$ with $t_{f,1} = 2.5\,\mathrm{s}$ and from $(0,0)^T$ to $(2,-1)^T$ with $t_{f,2} = 2\,\mathrm{s}$. The two trajectories were joined at $t_j = 0.6t_{f,1} = 1.5\,\mathrm{s}$. To obtain smooth blending of the goal positions $(x_g, z_g)^T$, the temporal scaling factor λ_t and the perturbation forces $f_{\approx,x}$, $f_{\approx,z}$, attractors of type (19) were used. For comparison, the trajectory generated using simple joining without blending is also shown in all plots (-).

stant parameters can also be used, although time-varying parameters produce smoother results. The initial conditions are $\boldsymbol{r}_g(0) = \boldsymbol{r}_{g,1}$ and $\dot{\boldsymbol{r}}_g(0) = \boldsymbol{0}$. System (19) is simulated for $0 < t_j < t_{f,1}$, while reproduction of the second DMP starts at t_j. Duration of the joined trajectory is therefore be $t_f = t_j + t_{f,2}$. Hence, we start blending the second trajectory before reproduction of the first one is finished.

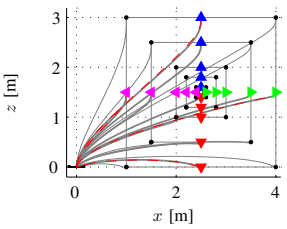
Fig. 6 shows the blended goal, DMP forces and z-coordinate for simple joining and proxy-based joining. Clearly, proxy-based joining produces a continuous result.

IV. Analysis and Experimental Results

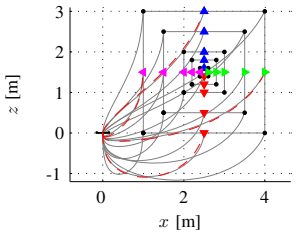
A. Trajectory generalization

In order to select an appropriate grid size for a given maneuver, we evaluated the DMP generalization error for various grid sizes (for the point-to-point and perching maneuvers). Figs 7(a) and 7(b) depict the evaluated generalized points for the two maneuvers. We chose the center point of the grid and edge midpoints where the expected errors are largest. An optimal solution was obtained for each of the evaluated points for comparison. The generalization error is shown in Fig. 7. We compare the cost function of the generalized trajectory to the optimal one. The increase is shown in relative terms, as $\tilde{\mathcal{J}}_i = (\mathcal{J}_{\mathrm{dmp},i} - \mathcal{J}_{\mathrm{opt}})/\mathcal{J}_{\mathrm{opt}}$. We additionally show the RMS of all considered points $\tilde{\mathcal{J}}_{\mathrm{all}}$ to evaluate the total generalization error for the grid size. Additionally, we provide the RMS error of the generalized (x, z) position trajectory w.r.t. the optimal one.

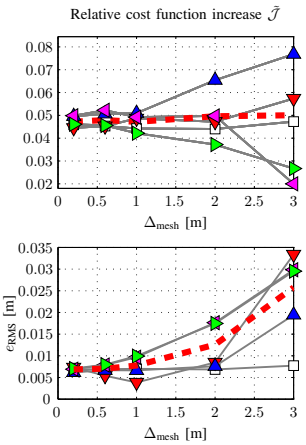
For both maneuvers the trajectory error e_{RMS} rises exponentially with grid size. However, the cost function is a better indicator of the generalization error. The point-to-point



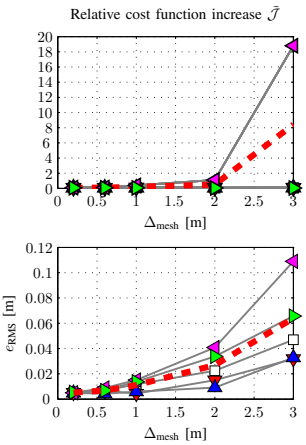
(a) Mesh for the point-to-point maneuver



(b) Mesh for the perching maneuver



(c) Point-to-point generalization error



(d) Perching generalization error

Fig. 7. Generalization error for the maneuvers. Figs. (a) and (b) show the trajectory shape from $(0,0)$ to different goal points. The markers represent the interpolation points. Figs. (c) and (d) show the generalization errors of the corresponding interpolation points for different grid sizes Δ_{mesh}. The thick dashed line in (c) and (d) shows the RMS of all trajectory errors $\tilde{\mathcal{J}}_{\mathrm{all}}$. For the point-to-point maneuver (a), the cost function of the generalized trajectories remains constant through various mesh size, even though the (x, z) trajectory error e_{RMS} rises exponentially. In contrast, the perching maneuver (b) is very sensitive to grid size. At 3 m grid size, the maximum cost function increase is 20-fold, for the leftmost generalized point. Accordingly, the RMS of all cost functions increases 9-fold. Here, the trajectory error e_{RMS} shows the same behaviour as in the point-to-point maneuver. Hence, smaller grid sizes should be used for highly dynamic maneuvers.

maneuver is mostly invariant to the grid size, as the cost function increase remains almost constant through all grid sizes, at about 5% above the optimal one. Generalization on the edges degrades with grid size, and generalization in the center of the grid remains almost constant. This could be explained by mostly polynomial control inputs for this type of maneuver, which can be nicely spatially scaled.

The perching maneuver shows exponential increase in the cost function, dominated by the error in the left grid edge. This indicates that the maneuver's generalization accuracy is very sensitive to grid size. The control inputs for the maneuver are nonlinear and obviously do not lend themselves to spatial scaling. Hence, for complex maneuvers a smaller grid size should be used.

B. Experimental and simulation results

The experiments were carried out using an AscTec Hummingbird quadrotor with a custom quaternion-based attitude controller with disturbance observation. The position and

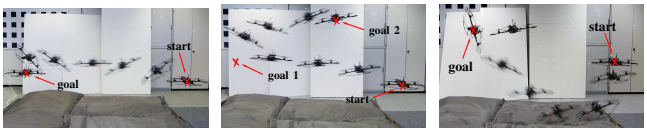
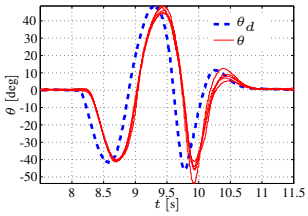
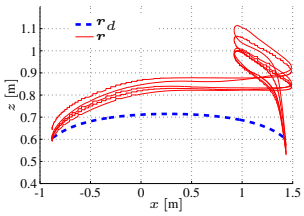


Fig. 8. Composite photos of the performed experiments that can be seen in the video attachment. From left to right: point-to-point maneuver; two joined point-to-point maneuvers; perching maneuver.



(a) Euler Angle θ.



(b) Reference $\boldsymbol{r}_d$ and actual trajectory $\boldsymbol{r}$ in the (x, z)-plane.

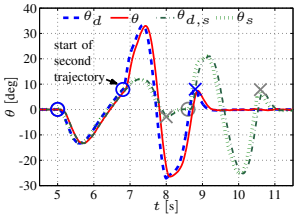
Fig. 9. Experimental results of five point-to-point maneuvers in minimum time. It can be seen that the model inaccuracies lead to large deviations for fast maneuvers. Optimal $\mathcal{J}_{\text{opt}} = 2.004 \times 10^3$, experimental $\mathcal{J}_{\text{exp}} = 3.036 \times 10^3$. During the maneuver, the velocity reaches 3.9 m/s at $t = 9.2$ s.

attitude are measured using an external A.R.T. tracking system running at 60 Hz. A marker is rigidly attached to the quadrotor frame. The IMU measurement range is limited to 300 °/s, therefore our trajectories were computed accordingly. The DMP reproduction and position controller are running in Simulink, with attitude and thrust commands sent wirelessly via XBee. The maneuvers were flown in the (x, z)-plane. Simulations were carried out in Simulink, using a full six-degrees of freedom quadrotor model.

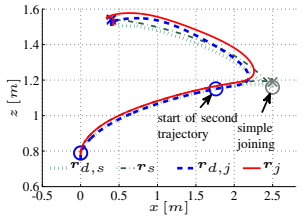
Fig. 8 depicts composite photos of the performed experiments. We show extensions to the DMP approach using the point-to-point maneuver. The perching maneuver was reproduced without perching mechanism, so the quadrotor was commanded to stabilize at the end of the trajectory.

Fig. 9 shows the pitch Euler angle and the (x, z)-trajectory during five point-to-point maneuvers. The limiting factor in the experiment is the angular velocity due to the IMU. The maximum angle is thus 45 ° during the maneuver. The trajectory shows large position overshoot at the end of the trajectory. Since the velocity reaches 3.9 m/s during the maneuver, unmodeled aerodynamics come into effect and diminish the tracking accuracy.

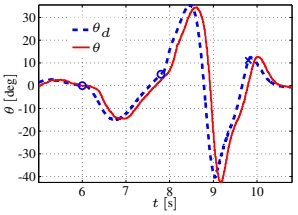
The simulation of trajectory joining in Fig. 10 shows the difference between simple joining (attaching two trajectories) and the joining method presented in this paper. Simple joining results in zero velocity at the joining point. Hence, the trajectories are just reproduced one after the other. The trajectory stops in the rightmost circle in Fig. 10(a). With our joining method, at $t_{0,2} = 0.6t_{f,1}$, the trajectory velocity is nonzero at the joining point. Larger accelerations are produced than in simple joining. However, the trajectory quickly converges to the reproduced one. The trajectory stops in the lefttmost circle in Fig. 10(a). Proxy-based joining clearly produces smooth trajectories, and leads to the same final position as simple joining. Fig. 10 depicts a proxy-based joining experiment. The same effect at the end of



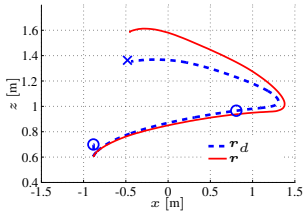
(a) Simulated Euler Angle θ.



(b) Simulated reference and trajectories for simple joining ($\boldsymbol{r}_{d,s}$, $\boldsymbol{r}_s$) and proxy-based joining ($\boldsymbol{r}_{d,j}$, $\boldsymbol{r}_j$).



(c) Experimental Euler Angle θ.



(d) Experimental reference and trajectory for proxy-based joining.

Fig. 10. Simulation and experimental results of joined and simple point-to-point maneuvers. The maneuvers were joined at $0.6t_{f,1}$ for proxy-based joining, and at $1.2t_{f,1}$ for simple joining. Circles mark the start of a trajectory, and the cross marks the end of a maneuver.

the trajectory can be seen as in the point-to-point maneuver, owing to unmodeled aerodynamics.

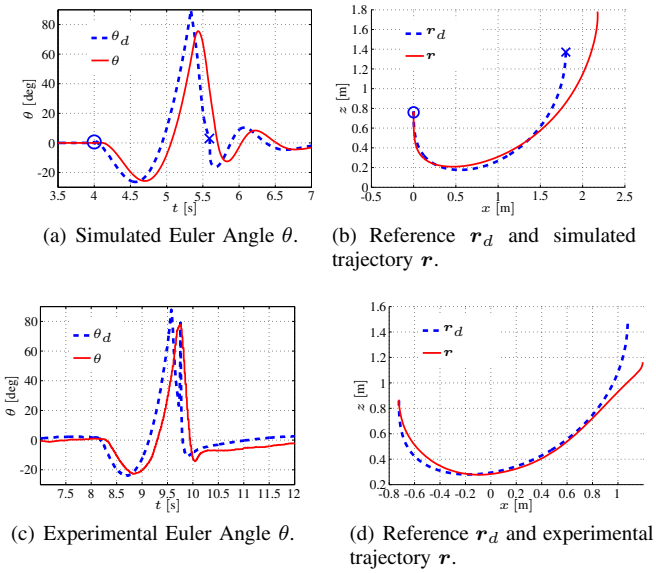
Fig. 11 shows simulation and experimental results for the perching maneuver. The attitude controller dynamics are not considered in the model used for trajectory generation. Therefore, the desired pitch angle of 90° was not reached exectly in simulation nor experiment. This indicates that either the controller dynamics have to be considered in the trajectory generation, or a method for improving trajectory execution is required.

C. A comment on computational complexity

We shortly outline the computational complexity for a grid of K trajectories, each approximated with N basis funtions, and sampled at M points. Offline, the optimal trajectory has to be computed K times, and a $KM \times N$ matrix must be pseudo-inverted. We obtain $K \times N$ weights to be stored. For online reproduction first the interpolation weights are obtained. The $\exp(\cdot)$ function must be evaluated N times for each integration step. Hence, the computational complexity during reproduction depends on the integration time step and number of basis functions.

V. Conclusion

In this paper we presented a novel method for learning optimal control solutions and generalizing them in real-time for a quadrotor-type vehicle with flat dynamics. The generalization is based on an adapted DMP approach. More specifically, our algorithm encodes a grid of optimal solutions that were generated offline into a dynamical system of second order. The trained DMP is then able to generalize to unforeseen goal states instantaneously via weight intrapolation. The effectiveness of the approach was shown in simulations and experiments.



(a) Simulated Euler Angle θ.

(b) Reference $\boldsymbol{r}_d$ and simulated trajectory $\boldsymbol{r}$.

(c) Experimental Euler Angle θ.

(d) Reference $\boldsymbol{r}_d$ and experimental trajectory $\boldsymbol{r}$.

Fig. 11. Simulation and experimental results of perching maneuver to $(1.8\text{ m}, 0\text{ m}, 0.6\text{ m})$. The circle marks start of the maneuver, and the cross marks its end. The attitude controller dynamics are not considered in the model used for trajectory generation. Therefore, the goal attitude $\theta_g = \frac{\pi}{2}$ was not reached in simulation nor experiment.

Analysis has shown that for rather complex maneuvers, such as perching, a smaller grid size should be used to obtain good generalization properties. The point-to-point maneuver generalization error was shown to be mostly invariant to grid size. Furthermore, we presented a real-time approach to trajectory joining by proxy-based blending of trajectory parameters. This makes it possible to seamlessly combine partial solutions to more sophisticated maneuvers.

Our next steps are to perform a thorough analysis of the limitations of the presented approach. We will also compare other DMP joining methods with the presented one. Furthermore, the approach presented in this paper is fully applicable to 6-DOF maneuvers, which we intend to validate.

Acknowledgments

This work has been partially funded by the European Commissions Sixth Framework Programme as part of the project SAPHARI (grant number 287513). The authors would like to thank R. Weitschat, F. Huber, R. Belder, G. Falconi, C. Heise and F. Holzapfel for helpful comments and discussions.

References

[1] T. Tomić, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 46 – 56, 2012.

[2] K. Schmid, T. Tomić, F. Rueß, H. Hirschmüller, and M. Suppa, "Stereo Vision Based Indoor/Outdoor Navigation for Flying Robots," in *IROS*, 2013.

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *ICRA*, 2011, pp. 2520 – 2525.

[4] J. Gillula, H. Huang, M. Vitus, and C. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *ICRA*, 2010, pp. 1649 – 1654.

[5] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," in *Proceedings of the IFAC World Congress*, 2011.

[6] ——, "Real-time trajectory generation for interception maneuvers with quadrocopters," in *IROS*, 2012, pp. 4979 – 4984.

[7] R. Ritz, M. Müller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *IROS*, 2012, pp. 4972 – 4978.

[8] M. Hehn, R. Ritz, and R. D'Andrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, vol. 33, no. 1-2, pp. 69 – 88, 2012.

[9] S. Haddadin, R. Weitschat, F. Huber, M. C. Oezparpucu, N. Mansfeld, and A. Albu-Schaeffer, "Optimal control for viscoelastic robots and its generalization in real-time," in *International Symposium on Robotics Research (ISRR2013)*, Singapore, 2013.

[10] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *ICRA*, 2002, pp. 1398 – 1403.

[11] S. Lupashin, A. Schoellig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *ICRA*, 2010, pp. 1642 – 1648.

[12] O. Purwin and R. D'Andrea, "Performing aggressive maneuvers using iterative learning control," in *ICRA*, 2009, pp. 1731 – 1736.

[13] R. Ritz, M. Hehn, S. Lupashin, and R. D'Andrea, "Quadrocopter performance benchmarking using optimal control," in *IROS*, 2011, pp. 5179 – 5186.

[14] H. Sira-Ramirez and S. Agrawal, *Differentially Flat Systems*. Marcel Dekker Inc, 2004.

[15] C. Darby, W. Hager, and A. Rao, "An hp-adaptive pseudospectral method for solving optimal control problems," *Optimal Control Applications and Methods*, vol. 32, pp. 476 – 502, 2010.

[16] A. Rao and D. Benson, *User's Manual for GPOPS Version 5.0: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Pseudospectral Methods*, August 2011.

[17] A. Rao, D. Benson, C. Darby, M. Patterson, C. Francolin, I. Sanders, and G. Huntington, "Algorithm 902: GPOPS, a MATLAB software for solving multiple-phase optimal control problems using the gauss pseudospectral method," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, p. 9, 2011.

[18] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *ICRA*, 2009, pp. 763 – 768.

[19] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," University of Southern California, Tech. Rep., 2003.

[20] R. Weitschat, S. Haddadin, , F. Huber, and A. Albu-Schäffer, "Dynamic optimality in real-time: A learning framework for near-optimal robot motions," in *IROS*, 2013.

[21] J. Kober, K. Mülling, O. Krömer, C. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *ICRA*, 2010, pp. 853 – 858.

[22] P. Kormushev, S. Calinon, and D. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *IROS*, 2010, pp. 3232 – 3237.

[23] B. Nemec, M. Zorko, and L. Zlajpah, "Learning of a ball-in-a-cup playing robot," in *Proceedings of the International Workshop on Robotics in Alpe-Adria-Danube Region*, 2010, pp. 297 – 301.

[24] S. Schaal, "Dynamic movement primitives a framework for motor control in humans and humanoid robotics," University of Southern California, Tech. Rep., 2003.

[25] A. Schoellig, M. Hehn, S. Lupashin, and R. DAndrea, "Feasiblity of motion primitives for choreographed quadrocopter flight," in *American Control Conference*, 2011, pp. 3843 – 3849.

[26] A. Schoellig, C. Wiltsche, and R. DAndrea, "Feed-forward parameter identification for precise periodic quadrocopter motions," in *American Control Conference*, 2012, pp. 4313 – 4318.

[27] A. Gams, T. Petric, L. Zlajpah, and A. Ude, "Optimizing parameters of trajectory representation for movement generalization: Robotic throwing," in *Proceedings of the International Workshop on Robotics in Alpe-Adria-Danube Region*, 2010, pp. 161 – 166.

[28] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Wörgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Transactions on Robotics*, vol. 28, pp. 145 – 157, 2012.

[29] B. Nemec and A. Ude, "Action sequencing using dynamic movement primitives," *Robotica*, vol. 30, pp. 837–846, 9 2012.

[30] M. V. Damme, B. Vanderborght, R. V. Ham, B. Verrelst, F. Daerden, and D. Lefeber, "Proxy-based sliding mode control of a manipulator actuated by pleated pneumatic artificial muscles," in *ICRA*, 2007, pp. 4355 – 4360.

[31] R. Kikuuwe, S. Yasukouchi, H. Fujimoto, and M. Yamamoto, "Proxy-based sliding mode control: A safer extension of PID position control," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 670 – 683, August 2010.