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 in terms of real-time generation of isolated 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 set of states. 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.
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 $\theta$ 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 $\tau$ and $J$ are the torque and inertia about the body $y$-axis, respectively. Using the state $x = [x \ 0 \ 0 \ \dot{\theta} \ \dot{\theta}]^T$ and control $u = [u_1 \ u_2]^T$, the equations of motion for the planar model are

$$
\begin{align*}
\ddot{x} & = u_1 \sin(\theta) \\
\dot{\theta} & = u_2.
\end{align*}
$$

(1)

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

$$
\begin{align*}
u_1 & = \sqrt{(\ddot{z}_d + g)^2 + \ddot{x}_d^2}, \\
\theta & = \text{atan2}(\ddot{x}_d, g + \ddot{z}_d),
\end{align*}
$$

(2)

where $\text{atan2}$ 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 $u^*$ which causes the system

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

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

$$
x_{min} \leq x(t) \leq x_{max}, \quad u_{min} \leq u(t) \leq u_{max},
$$

and the boundary conditions $x(t_0) = x_0, x(t_f) = x_f$, to follow a trajectory $x^*$ that minimizes the optimality criterion

$$
J(x(t), u(t), t) = \mathcal{J}_M(x(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{J}_L(x(t), u(t), t) \, dt.
$$

Model (1) can be rewritten as

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

(4)

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

$$
\frac{T}{m} \leq u_1 \leq \frac{T}{m}, \quad |u_2| \leq \frac{\tau}{J},
$$

(5)

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

For the purposes of learning maneuvers, 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

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

(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-
\section*{TABLE I}

\textbf{SYSTEM PARAMETERS, STATE AND INPUT CONSTRAINTS USED FOR THE}

\textbf{OPTIMAL CONTROL PROBLEM AND THE PLANAR QUADROTOR MODEL.}

<table>
<thead>
<tr>
<th>System parameters</th>
<th>State constraints</th>
<th>Input constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m = 0.5 \text{ kg}$</td>
<td>$\tau = 10 \text{ m/s}$</td>
<td>$T = 12 \text{ N}$</td>
</tr>
<tr>
<td>$J = 3 \times 10^{-3} \text{ kgm}^2$</td>
<td>$\dot{\tau} = -10 \text{ m/s}$</td>
<td>$T = 1 \text{ N}$</td>
</tr>
<tr>
<td>$g = 9.81 \text{ m/s}^2$</td>
<td>$\dot{\tau} = 300 \text{ Nm/s}$</td>
<td>$\tau = 0.2 \text{ Nm}$</td>
</tr>
<tr>
<td>$\theta = -300 \text{ °/s}$</td>
<td>$\tau = -0.2 \text{ Nm}$</td>
<td></td>
</tr>
</tbody>
</table>

\section*{TABLE II}

\textbf{OVERVIEW OF MANEUVERS CONSIDERED IN THIS PAPER}

<table>
<thead>
<tr>
<th>Maneuver</th>
<th>Via condition</th>
<th>End condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point-to-point</td>
<td>–</td>
<td>$p_f = [x_f, y_f, 0]^T$, $p_f = [0, 0, 0]^T$</td>
</tr>
<tr>
<td>Perching</td>
<td>–</td>
<td>$p_f = [x_f, y_f, z_f]^T$, $p_f = [0, 0, 0]^T$</td>
</tr>
<tr>
<td>Flip</td>
<td>$p_{via} = [x_{via}, z_{via}, \pi]^T$</td>
<td>$p_f = [x_f, 0, 0]^T$, $p_f = [0, 0, 0]^T$</td>
</tr>
</tbody>
</table>

For notational simplicity we will define the maneuver constraints through the position $p = [x \ z \ \theta]^T$ and velocity $\dot{p} = [\dot{x} \ \dot{z} \ \dot{\theta}]^T$. All maneuvers start at initial conditions $p_0 = [0 \ 0 \ 0]^T$, $\dot{p}_0 = [0 \ 0 \ 0]^T$, with the via and end conditions specified in Table II. Fig. 2 shows the trajectories of three maneuvers considered in this paper.

\subsection*{a) Point-to-point maneuver:} The goal is to reach a desired location $p_f = [x_f, y_f, 0]^T$ in minimum time. Zero velocity is imposed at start and end of the maneuver.

\subsection*{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.

\subsection*{c) Flip:} During the maneuver, the quadrotor passes a specific via point with the attitude of $\theta_{via} = \pi$ (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 $p_f$. We obtain a set of optimal trajectories, which are learned using a representation that allows generalization to new goals.

\section*{III. MANEUVER LEARNING AND GENERALIZATION}

\subsection*{A. Dynamic Movement Primitives}

Dynamic Movement Primitives (DMPs) were 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 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 rhythmic 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

$$m \ddot{r} + d \dot{r} + \kappa (r - r_g) + f_t = 0,$$

(7)

where $r = [x \ y \ z]^T$ 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 $f_t = [f_{tx} f_{ty} f_{tz}]^T$ 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 $f_t$. 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 $r$, velocity $\dot{r}$, and acceleration $\ddot{r}$ using only the learned force $f_t$. The goal position $r_g$ allows the generalization of the movement primitive to new goals. For $f = 0$, it can be shown that $r_g$ is the asymptotically BIBO stable equilibrium.

To learn and approximate the perturbation force $f_t$ which
is needed to produce a desired trajectory \( r(t) \) using the mass-spring-damper system (7), \( f_t \) is encoded into a normalized Radial Basis Function (RBF) with \( N \) basis functions [10] that is defined as

\[
f_s(s) = \frac{\sum_{i=1}^{N} w_i \Psi_i(s)}{\sum_{i=1}^{N} \Psi_i(s)} s,
\]

where the Gaussian basis is defined as

\[
\Psi_i(s) = \exp \left( -h_i(s-c_i)^2 \right).
\]

Its center points are located at

\[
c_i = \exp \left( -\alpha i \frac{i-1}{N-1} \right).
\]

The widths are defined as

\[
h_i = \left\{ \begin{array}{ll}
\frac{\beta_i}{(c_{i+1} - c_i)} + \gamma_s, & i = 1, \ldots, N-1, \\
\frac{1}{2}, & i = N.
\end{array} \right.
\]

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

\[
\dot{s} = -\alpha s, \quad s(0) = 1, \quad s(t_f) = 0.
\]

Definition of the DMP through \( s \) enables temporal scaling of the trajectories. Furthermore, as \( s \) decreases towards zero, the perturbation force \( f_s \) 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 \( \beta_i \neq 0 \) 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

\[
A w = f_s \approx f_t,
\]

where

\[
A = \begin{bmatrix}
\Psi_1(s_1) & \ldots & \Psi_N(s_1) & s_1 \\
\sum_{i=1}^{N} \Psi_i(s_1) & \ldots & \sum_{i=1}^{N} \Psi_i(s_1) & 1 \\
\Psi_1(s_M) & \ldots & \Psi_N(s_M) & s_M \\
\sum_{i=1}^{N} \Psi_i(s_M) & \ldots & \sum_{i=1}^{N} \Psi_i(s_M) & 1
\end{bmatrix}.
\]

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

\[
\Gamma = \sum_{i=1}^{M} (f_{t,i} - f_{s,i})^2
\]
can thus be optimized by the left Moore-Penrose pseudo-inverse, i.e. linear regression

\[
w^* = A^+ f_t.
\]

The quality of the approximation depends significantly on \( \alpha_s, \beta_s, \gamma_s, \hat{\kappa}, \) 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 \( \alpha_s, \beta_s, \gamma_s \), that minimizes the goal function \( \Gamma \), was done using the Matlab function \( \text{fmincon} \). 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 \( w^*_s \) of the Gaussian basis, the learned trajectory can be reproduced by numerically integrating

\[
\ddot{m}\ddot{r} = -\ddot{\hat{\kappa}}(t)(\dot{r} - \dot{r}_g) - f_s,
\]

with initial conditions \( \dot{\hat{\kappa}}(0) = \hat{\kappa}_0, \ddot{\hat{\kappa}}(0) = \ddot{r}_0 \). Here, \( \ddot{\hat{\kappa}}(t) \) and \( \hat{\kappa}(t) \) are time-varying damping and stiffness defined below.

We use the approximated perturbation force \( f_s \) as an input. For reproduction, we note that the trajectory will reach the goal also for nonzero initial conditions \( r_0 \) and \( \dot{r}_0 \). 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 \( \kappa(x(t = 0) - x_g) \) in (7), which produces a jump in the acceleration at the start of the trajectory. We use the stiffness

\[
\hat{\kappa}(t) = \kappa \left( 1 - \exp \left( -\frac{t}{k t_f} \right) \right),
\]

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

Set
- mass $m$,
- number of Gaussians $N$,
- parameters for time-varying stiffness $\dot{\kappa}(t)$,
- initial parameters $\alpha^0$, $\beta^0$, $\gamma^0$.

Compute
- time-varying stiffness $\dot{\kappa}(t)$ and damping $\dot{d}(t)$.

Output: optimal weights $w^*$
and parameters $\alpha^*$, $\beta^*$, $\gamma^*$.

Converged or $\Gamma$ below error tolerance?

Optimization loop

(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 $r_g$ from $r_{g,1}$ to $r_{g,2}$ using the system

$$m_p\ddot{r}_g + \dot{d}_p r_g + \dot{\kappa}_p (r_g - r_{g,2}) = 0,$$

where $m_p$ is the proxy mass, $\dot{\kappa}_p$ is the time-varying proxy stiffness and $\dot{d}_p$ is the time-varying proxy damping. Con-
constant parameters can also be used, although time-varying parameters produce smoother results. The initial conditions are \( r_j(0) = 0 \) and \( \dot{r}_j(0) = 0 \). System (19) is simulated for \( 0 < t_j < t_{f,j} \), 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 \( \tilde{J}_i = (J_{\text{DMP},i} - J_{\text{opt}}) / J_{\text{opt}} \). We additionally show the RMS of all considered points \( \tilde{J}_{\text{all}} \) 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 \( e_{\text{RMS}} \) rises exponentially with grid size. However, the cost function is a better indicator of the generalization error. The point-to-point 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
The effectiveness of the approach was shown in simulations and experiments. It can be seen that the model inaccuracies lead to large deviations in the video attachment. From left to right: point-to-point maneuver; two joined point-to-point maneuvers; perching maneuver.

Fig. 8 depicts 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.

Fig. 9 shows the pitch Euler angle and the \((x, z)\)-trajectory in the \((x, z)\)-plane. Simulations were carried out in Simulink, using a full six-degrees of freedom quadrotor model.

Fig. 10 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 exactly 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 functions, 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 interpolation. The effectiveness of the approach was shown in simulations and experiments.
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


