2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

An On-Board Learning Scheme for Open-Loop Quadrocopter Maneuvers Using Inertial Sensors and Control Inputs from an External Pilot

Robin Ritz and Raffaello D'Andrea

***Abstract*— We present an iterative learning scheme for improving the performance of highly dynamic open-loop maneuvers with quadrocopters. A probabilistic estimate of the state deviation at the end of the maneuver is obtained by fusing two data sources that are available on-board: 1) an inertial measurement unit, and 2) control inputs from an external pilot that performs a recovery after the open-loop maneuver has been executed. A computationally lightweight policy gradient method is applied in order to adapt a set of characteristic maneuver parameters, which in turn reduces the expected value of the final state deviation for the next execution of the maneuver. The performance of the learning algorithm is demonstrated in the ETH Zurich Flying Machine Arena by improving the performance of a triple flip.**

I. Introduction

In recent years, the growing popularity of the quadrocopter as a research platform has led to several demonstrations of its ability to perform highly dynamic maneuvers (examples include [1], [2], and [3]). For most of these maneuvers, a first-principles model is leveraged to plan the trajectories, and subsequently a feedback control law is applied for tracking them. While for most tasks it is sufficient to apply a feedback controller in order to compensate for the unmodeled effects, for highly dynamic maneuvers these effects may become significant, which results in large tracking errors. This problem is not limited to quadrocopters, but arises in many dynamical systems when performing aggressive tasks. A common approach to improve tracking performance in such situations is to execute the maneuver of interest multiple times and then apply an iterative learning method that non-causally corrects for systematic errors. In other words, data from previous executions is used to adapt the nominal maneuver parameters, such that in the ideal case all repeatable disturbances are eliminated after a sufficient number of iterations.

One well-known approach for adapting a maneuver iteratively is called iterative learning control [4], [5], [6]: A correction term is introduced to each time step of a discrete-time representation of the desired trajectory, where the correction terms typically act on the reference inputs or on the nominal state trajectory. After each execution of the maneuver, an optimization over the correction terms is performed in order to minimize some measure of the expected tracking error. Various learning strategies based on this approach have been introduced, for example enabling more precise motions of robot arms [7], or improving tracking performance with quadrocopters [8].

A second approach for correcting systematic disturbances is to adapt only a small set of characteristic parameters [9], [10]. This strategy is typically suitable if the objective is to minimize the deviation from the nominal maneuver only at certain key frames, rather than at every time step of the trajectory. Due to the lower dimensionality of the problem this approach usually requires less computational resources, which is favorable for on-board implementation. Therefore, the learning scheme presented herein applies this approach.

A crucial requirement for most learning strategies is the ability to measure the deviation from the nominal trajectories. Typically, learning methods for high-performance quadrocopter maneuvers leverage an external motion-capture system to precisely measure the vehicle's position and attitude at a high rate, which significantly simplifies the estimation of the state deviation [8], [9], [10], [11]. However, a motion-capture system is not always available or practical, and many applications require other approaches for estimating the vehicle's state deviation. In this paper, we extend the parameter adaptation method introduced in [9] to a scenario where no position and attitude measurements are available to the learning algorithm. The learning strategy is computationally lightweight and implemented on-board. We assume that an external pilot is controlling the vehicle both before and after the open-loop maneuver. The data sources available to the learning algorithm are: 1) an inertial measurement unit, and 2) the control commands that are received from the external pilot before and after the execution of the maneuver.

The remainder of this paper is structured as follows: In Section II, we present a first-principles model of the quadrocopter. Section III introduces an estimator that provides a probabilistic measure of the state offset after the open-loop maneuver, and Section IV presents an iterative learning algorithm for improving performance. In Section V we show experimental results, and we conclude in Section VI.

II. Model

This section presents a first-principles model of the quadrocopter, where we use the model described in [9]. The vehicle is modeled as a rigid body with mass m and rotational inertia $\boldsymbol{I}$ around the center of gravity. The position $\boldsymbol{p} = (p_x, p_y, p_z)$ of the vehicle, which also refers to

This research was supported by the Hans-Eggenberger Stiftung and the SNSF (Swiss National Science Foundation). The authors are with the Institute for Dynamic Systems and Control, ETH Zürich.
{rritz, rdandrea}@ethz.ch

its center of gravity, is measured in an inertial frame O. (For the ease of notation, throughout this paper vectors may be expressed as n-tuples $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, with dimension and stacking clear from context.) We choose the inertial frame O such that gravity is aligned with the negative z-direction. The attitude of the vehicle is described by a rotation matrix $\boldsymbol{R}$ that represents a rotation from the body frame B (which is fixed on the vehicle) to the inertial frame O.

A. Vehicle Dynamics

The vehicle provides four motors with fixed-pitch propellers as actuators, each of them producing a thrust force $f_p,\ p \in \{1,2,3,4\}$. Let $\boldsymbol{a} = (a_x, a_y, a_z)$ be the proper acceleration[1] in the body frame B. All propeller forces f_p act along the body z-axis, and hence the proper acceleration $\boldsymbol{a}$ is given by

$$\boldsymbol{a} = \begin{bmatrix} 0 \\ 0 \\ (f_1 + f_2 + f_3 + f_4)/m \end{bmatrix} + \tilde{\boldsymbol{a}}_{model}, \tag{1}$$

where $\tilde{\boldsymbol{a}}_{model}$ denotes the process noise acting on the acceleration. The translational dynamics are given by the second order differential equation

$$\ddot{\boldsymbol{p}} = \boldsymbol{R}\boldsymbol{a} + \boldsymbol{g}, \tag{2}$$

where $\boldsymbol{g} = (0, 0, -g)$ denotes the gravitational acceleration. The rotation matrix $\boldsymbol{R}$ evolves with

$$\dot{\boldsymbol{R}} = \boldsymbol{R} \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{3}$$

where $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ denotes the rotational rates around the vehicle's body axes. The dynamics of the body rates $\boldsymbol{\omega}$ are governed by

$$\dot{\boldsymbol{\omega}} = \boldsymbol{I}^{-1} \left(\begin{bmatrix} d(f_2 - f_4) \\ d(f_3 - f_1) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix} - \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \right) + \tilde{\boldsymbol{\omega}}_{model}, \tag{4}$$

where d denotes the arm length of the quadrocopter, κ the torque-to-thrust ratio of the propellers, and $\tilde{\boldsymbol{\omega}}_{model}$ the process noise acting on the rotational acceleration.

B. Sensor Model

An inertial measurement unit is mounted on the vehicle, measuring the proper acceleration $\boldsymbol{a}$ and the body rate $\boldsymbol{\omega}$. The acceleration measurement is modeled as

$$\boldsymbol{a}_{meas} = \boldsymbol{a} + \tilde{\boldsymbol{a}}_{meas}, \tag{5}$$

where $\boldsymbol{a}_{meas}$ is the sensor output value, and $\tilde{\boldsymbol{a}}_{meas}$ denotes the sensor noise. Similar to the acceleration, the angular rate measurement is modeled as

$$\boldsymbol{\omega}_{meas} = \boldsymbol{\omega} + \tilde{\boldsymbol{\omega}}_{meas}, \tag{6}$$

with $\boldsymbol{\omega}_{meas}$ being the sensor output, and $\tilde{\boldsymbol{\omega}}_{meas}$ denoting the sensor noise.

[1] In this context, *proper acceleration* refers to the acceleration relative to free fall.

C. On-board Controller

The vehicle receives desired total thrust f_{tot}^{des} and desired body rates $\boldsymbol{\omega}^{des}$ from an external pilot or from the open-loop maneuver definition. The on-board controller is designed such that the measured body rate deviations $\Delta\boldsymbol{\omega} = \boldsymbol{\omega}^{des} - \boldsymbol{\omega}_{meas}$ follow three decoupled first-order systems. This is achieved by inverting the angular dynamics (4), while the process noise $\tilde{\boldsymbol{\omega}}_{model}$ is neglected:

$$\begin{bmatrix} d(f_2 - f_4) \\ d(f_3 - f_1) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix} = \boldsymbol{I} \begin{bmatrix} \Delta\omega_x/\tau_{xy} \\ \Delta\omega_y/\tau_{xy} \\ \Delta\omega_z/\tau_z \end{bmatrix} + \boldsymbol{\omega}_{meas} \times \boldsymbol{I}\boldsymbol{\omega}_{meas}, \tag{7}$$

where τ_{xy} is the desired time constant for the x- and y-axis, and τ_z the time constant around the z-axis. The above equation, together with the condition

$$f_{tot}^{des} = f_1 + f_2 + f_3 + f_4, \tag{8}$$

defines the four thrust forces f_p.

D. Two-dimensional Vehicle Dynamics

In this paper, we apply the proposed learning scheme to a two-dimensional open-loop maneuver (namely a triple flip that we will introduce in Section V). To reduce the computational complexity, we thus restrict the learning to two dimensions and assume that the systematic out-of-plane disturbances are small. While in principle the approach presented herein is also suitable for three-dimensional learning, it is beyond the scope of this paper to verify this experimentally.

In the following, we reduce our model to the xz-plane. In doing so, the vehicle's position is described by p_x and p_z, and the attitude can be described by a single parameter θ, which is the pitch angle. Fig. 1 shows an illustration of the two-dimensional model.

For the two-dimensional case, the translational dynamics (2) reduce to

$$\ddot{p}_x = a_x \cos\theta + a_z \sin\theta, \tag{9}$$

$$\ddot{p}_z = -a_x \sin\theta + a_z \cos\theta - g. \tag{10}$$

The evolution of the attitude is simply

$$\dot{\theta} = \omega_y, \tag{11}$$

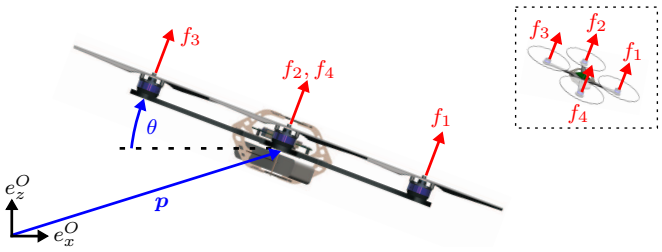

Fig. 1. Illustration of the two-dimensional quadrocopter model considered in this paper. The red arrows denote the propeller forces f_p, and the blue arrows show state vector components (position $\boldsymbol{p} = (p_x, p_z)$ and pitch angle θ). For clarification, a three-dimensional view is shown on the top right.

and the angular acceleration (4) for the two-dimensional model is given by

$$\dot{\omega}_y = d(f_3 - f_1)/I_{yy} + \tilde{\dot{\omega}}_{y,model}, \tag{12}$$

where we assume a diagonal rotational inertia matrix $\boldsymbol{I}$ with entry I_{yy} for the y-axis.

We define the state vector of the two-dimensional system to be

$$\boldsymbol{s} := (p_x, p_z, v_x, v_z, \theta), \tag{13}$$

where $v_x := \dot{p}_x$ and $v_z := \dot{p}_z$ denote the translational velocities. Note that the pitch rate ω_y is not included in the state vector, even though it is a dynamic state. This is because: 1) the pitch rate ω_y is estimated separately (as we will see in Section III), and 2) we do not intend to correct for pitch rate offsets after the open-loop maneuver, since we assume that the on-board controller (7) quickly controls these deviations to zero.

The current proper acceleration a_x and a_z, and the current pitch rate ω_y can be considered as inputs to the system describing the evolution of the state vector (13). Hence, we define an input vector

$$\boldsymbol{u} := (a_x, a_z, \omega_y), \tag{14}$$

that contains these variables. Stacking (9), (10), and (11) into a vector, the continuous-time dynamics $\boldsymbol{f}_c(\boldsymbol{s}, \boldsymbol{u})$ of the state vector (13) yield

$$\dot{\boldsymbol{s}} = \boldsymbol{f}_c(\boldsymbol{s}, \boldsymbol{u}) = \begin{bmatrix} v_x \\ v_z \\ a_x \cos\theta + a_z \sin\theta \\ -a_x \sin\theta + a_z \cos\theta - g \\ \omega_y \end{bmatrix}. \tag{15}$$

E. Discrete-time System Dynamics

Since the estimator will be designed for discrete time steps, the system dynamics (15) must be discretized. We use a superscript k to indicate the time step of a variable. The discrete-time system dynamics $\boldsymbol{f}_d(\boldsymbol{s}^k, \boldsymbol{u}^k)$ are approximated by

$$\boldsymbol{s}^{k+1} = \boldsymbol{f}_d(\boldsymbol{s}^k, \boldsymbol{u}^k) \approx \boldsymbol{s}^k + \boldsymbol{f}_c(\boldsymbol{s}^k, \boldsymbol{u}^k)T, \tag{16}$$

where T denotes the sampling time.

F. Jacobian Matrices for Covariance Matrix Prediction

The estimator which will be introduced in Section III leverages Jacobian matrices around the current estimated state for approximating the evolution of the covariance matrix. (This is similar to the prediction step of an extended Kalman filter.) Omitting the superscript k for reasons of readability, the Jacobian matrices $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ of the discrete-time system dynamics (16) around an arbitrary operating point ($\boldsymbol{s} = \boldsymbol{s}^*$, $\boldsymbol{u} = \boldsymbol{u}^*$) yield

$$\begin{aligned} \boldsymbol{A}_d(\boldsymbol{s}^*, \boldsymbol{u}^*) &= \left.\frac{\partial \boldsymbol{f}_d(\boldsymbol{s}, \boldsymbol{u})}{\partial \boldsymbol{s}}\right|_{\boldsymbol{s}=\boldsymbol{s}^*, \boldsymbol{u}=\boldsymbol{u}^*} \\ &= \begin{bmatrix} 1 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & -Ta_x^* \sin\theta^* + Ta_z^* \cos\theta^* \\ 0 & 0 & 0 & 1 & -Ta_x^* \cos\theta^* - Ta_z^* \sin\theta^* \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{17}$$

$$\begin{aligned} \boldsymbol{B}_d(\boldsymbol{s}^*, \boldsymbol{u}^*) &= \left.\frac{\partial \boldsymbol{f}_d(\boldsymbol{s}, \boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{\boldsymbol{s}=\boldsymbol{s}^*, \boldsymbol{u}=\boldsymbol{u}^*} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ T\cos\theta^* & T\sin\theta^* & 0 \\ -T\sin\theta^* & T\cos\theta^* & 0 \\ 0 & 0 & T \end{bmatrix}. \end{aligned} \tag{18}$$

This completes the derivation of the quadrocopter model used in this paper.

III. Estimator

In this section, we introduce an estimator that provides a probabilistic measure of the state evolution, relative to an initial state estimate.

A. Overview

An overview of the estimator is shown in Fig. 2. The estimator draws information from two different sources (shown as green boxes in Fig. 2). The first source is a first-principles model of the vehicle, which computes an estimate of the current translational and angular acceleration based on the four propeller forces that are currently applied. The second source of information is an inertial sensor, which measures acceleration and angular rate directly. Data fusion algorithms are used to merge this information, and subsequently position, velocity, and pitch angle estimates are obtained by numerical integration of the system dynamics.

In the following, all probabilistic variables are approximated by Gaussian distributions, and are thus defined by an expected value (denoted with a hat symbol $\hat{\cdot}$) and a covariance matrix (denoted as $\Sigma[\cdot]$). All noise terms are assumed to be Gaussian white noise signals [12]. The estimator is similar for a two-dimensional and for a three-dimensional model, except that the state and input vectors have different sizes. We use the more general three-dimensional notation when we state the estimator's equations in the following.

B. Acceleration Estimation

For each time step k we estimate the current proper acceleration $\boldsymbol{a}_{est}^k$ by fusing the model-based proper acceleration $\boldsymbol{a}_{model}^k$ (which is the the model-based acceleration $\boldsymbol{a}$ given by (1) evaluated at time step k) with the sensor measurement $\boldsymbol{a}_{meas}^k$, given by (5). Applying the assumption that there is no correlation between the noise terms of the two estimates, the covariance matrix $\Sigma[\boldsymbol{a}_{est}^k]$ of the fused acceleration can be written as

$$\Sigma[\boldsymbol{a}_{est}^k] = (\Sigma[\tilde{\boldsymbol{a}}_{model}^k]^{-1} + \Sigma[\tilde{\boldsymbol{a}}_{meas}^k]^{-1})^{-1}, \tag{19}$$

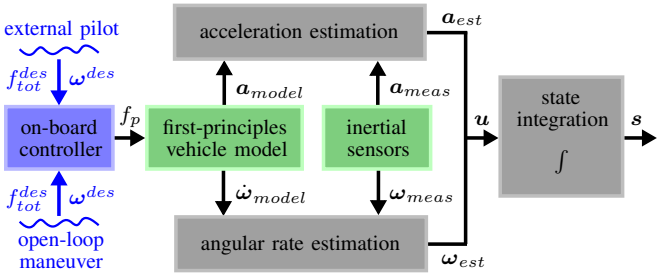

Fig. 2. Flowchart of the state estimator presented herein. The green blocks are the two data sources: the first-principle vehicle model and the inertial sensors. Note that, except for the external commands (f_{tot}^{des} and $\boldsymbol{\omega}^{des}$) and the propeller forces f_p, all signals have a probabilistic representation, and thus are approximated by a mean value and a covariance matrix. The on-board controller (blue block) takes commands either from an external pilot or from the open-loop maneuver description in order to compute the deterministic propeller forces f_p.

where $\tilde{\boldsymbol{a}}_{model}^k$ and $\tilde{\boldsymbol{a}}_{meas}^k$ denote the noise terms defined in (1) and (5), respectively. (We assume that the noise properties $\Sigma[\tilde{\boldsymbol{a}}_{model}^k]$ and $\Sigma[\tilde{\boldsymbol{a}}_{meas}^k]$ are known.) The expected value $\hat{\boldsymbol{a}}_{est}^k$ yields

$$\hat{\boldsymbol{a}}_{est}^k = \Sigma[\boldsymbol{a}_{est}^k](\Sigma[\tilde{\boldsymbol{a}}_{model}^k]^{-1}\hat{\boldsymbol{a}}_{model}^k + \Sigma[\tilde{\boldsymbol{a}}_{meas}^k]^{-1}\hat{\boldsymbol{a}}_{meas}^k), \tag{20}$$

where $\hat{\boldsymbol{a}}_{model}^k$ is the expected value of (1) and $\hat{\boldsymbol{a}}_{meas}^k$ is the acceleration sensor reading at time step k. Note that if the noise terms $\tilde{\boldsymbol{a}}_{model}^k$ and $\tilde{\boldsymbol{a}}_{meas}^k$ are modeled to be uncorrelated between x-, y-, and z-components, then (19) and (20) both decouple into scalar equations, one for each axis.

C. Angular Rate Estimation

In order to find an estimate of the current body rate $\boldsymbol{\omega}_{est}^k$, we fuse the model-based prediction of the angular acceleration (4), denoted here as $\dot{\boldsymbol{\omega}}_{model}^k$, with the current sensor reading of the rate gyro (5), denoted as $\boldsymbol{\omega}_{meas}^k$. In contrast to the acceleration, the body rate is a dynamic state and its model-based prediction at time step k therefore depends on the previous estimated value $\boldsymbol{\omega}_{est}^{k-1}$. The model-based prediction of the body rate $\boldsymbol{\omega}_{model}^k$ can be approximated by

$$\hat{\boldsymbol{\omega}}_{model}^k = \hat{\boldsymbol{\omega}}_{est}^{k-1} + \hat{\dot{\boldsymbol{\omega}}}_{model}^{k-1} T, \tag{21}$$

$$\Sigma[\boldsymbol{\omega}_{model}^k] = \Sigma[\boldsymbol{\omega}_{est}^{k-1}] + \Sigma[\tilde{\dot{\boldsymbol{\omega}}}_{model}^{k-1}]T^2, \tag{22}$$

where $\hat{\dot{\boldsymbol{\omega}}}_{model}^{k-1}$ is the expected value of (4) evaluated at the previous time step, and $\tilde{\dot{\boldsymbol{\omega}}}_{model}^{k-1}$ the corresponding process noise term with known covariance matrix $\Sigma[\tilde{\dot{\boldsymbol{\omega}}}_{model}^{k-1}]$. Again assuming no correlation between the noise terms of the model-based and sensor-based estimates, the fused angular rate estimate $\boldsymbol{\omega}_{est}^k$ yields

$$\hat{\boldsymbol{\omega}}_{est}^k = \Sigma[\boldsymbol{\omega}_{est}^k](\Sigma[\boldsymbol{\omega}_{model}^k]^{-1}\hat{\boldsymbol{\omega}}_{model}^k + \Sigma[\tilde{\boldsymbol{\omega}}_{meas}^k]^{-1}\hat{\boldsymbol{\omega}}_{meas}^k), \tag{23}$$

$$\Sigma[\boldsymbol{\omega}_{est}^k] = (\Sigma[\boldsymbol{\omega}_{model}^k]^{-1} + \Sigma[\tilde{\boldsymbol{\omega}}_{meas}^k]^{-1})^{-1}, \tag{24}$$

where $\hat{\boldsymbol{\omega}}_{meas}^k$ is the rate gyro reading at time step k, and $\tilde{\boldsymbol{\omega}}_{meas}^k$ the corresponding sensor noise with given covariance matrix $\Sigma[\tilde{\boldsymbol{\omega}}_{meas}^k]$.

D. State Prediction

Based on the acceleration estimate $\boldsymbol{a}_{est}^k$ and the body rate estimate $\boldsymbol{\omega}_{est}^k$ from above, we perform a prediction in order to obtain position, velocity, and pitch angle estimates, i.e. an estimate of the state vector $\boldsymbol{s}^k$ defined by (13). The prediction of the state's expected value $\hat{\boldsymbol{s}}^{k+1}$ is performed with

$$\hat{\boldsymbol{s}}^{k+1} = \boldsymbol{f}_d(\hat{\boldsymbol{s}}^k, \hat{\boldsymbol{u}}^k), \tag{25}$$

where the discretized state dynamics $\boldsymbol{f}_d(\hat{\boldsymbol{s}}^k, \hat{\boldsymbol{u}}^k)$ are given by (16). The evolution of the covariance matrix of the estimated state can be approximated by

$$\Sigma[\boldsymbol{s}^{k+1}] = \boldsymbol{A}_d^k \Sigma[\boldsymbol{s}^k](\boldsymbol{A}_d^k)^T + \boldsymbol{B}_d^k \Sigma[\boldsymbol{u}^k](\boldsymbol{B}_d^k)^T, \tag{26}$$

where for reasons of readability the shorthand notation $\boldsymbol{A}_d^k = \boldsymbol{A}_d(\hat{\boldsymbol{s}}^k, \hat{\boldsymbol{u}}^k)$ and $\boldsymbol{B}_d^k = \boldsymbol{B}_d(\hat{\boldsymbol{s}}^k, \hat{\boldsymbol{u}}^k)$ is used [13].

This completes the description of the state estimator. At every time step k, the expected value of the estimated state is updated with (25), and its covariance matrix is updated with (26). Note that since we have no feedback on the state vector $\boldsymbol{s}^k$, the expected value $\hat{\boldsymbol{s}}^k$ drifts away from the true value and the covariance $\Sigma[\boldsymbol{s}^k]$ grows over time.

IV. Learning Algorithm

In this section, we describe an iterative learning algorithm for improving the performance of a highly dynamic open-loop maneuver. The presented method is based on the policy gradient learning scheme introduced in [9]. A first-principles model of the vehicle is used for computing an initial guess of the maneuver parameters, and the same model is leveraged to derive an error correction matrix used for iteratively improving these maneuver parameters.

The initial parameter guess and the error correction matrix are computed in advance using a desktop workstation, while the parameter improvement scheme is running on-board.

A. Initial Parameter Guess

We assume that the open-loop maneuver to be executed has a defined structure, hence we can select a set of N numerical parameters

$$\boldsymbol{p} = (p_1, \ldots, p_N), \tag{27}$$

that define the desired maneuver trajectories

$$f_{tot}^{des}(t),\ \boldsymbol{\omega}^{des}(t),\ \text{for } t \in [t_0, t_f], \tag{28}$$

with t_0 and t_f denoting the start and end time of the open-loop maneuver, respectively. We model the estimate of the maneuver parameters $\boldsymbol{p}$ to be a Gaussian distribution with expected value $\hat{\boldsymbol{p}}$ and covariance matrix $\Sigma[\boldsymbol{p}]$. For a given initial state $\hat{\boldsymbol{s}}_0$, the nominal final state $\hat{\boldsymbol{s}}_f^{nom}(\hat{\boldsymbol{p}})$ is obtained by integrating the nominal system dynamics:

$$\hat{\boldsymbol{s}}_f^{nom}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{s}}_0 + \int_{t_0}^{t_f} \dot{\boldsymbol{s}}^{nom}(t, \hat{\boldsymbol{p}})\mathrm{d}t. \tag{29}$$

The nominal dynamics $\dot{\boldsymbol{s}}^{nom}(t, \hat{\boldsymbol{p}})$ are given by (15), where process noise terms are neglected.[2] In order to find an initial

[2]The argument $(\hat{\boldsymbol{p}})$ is stated explicitly in order to highlight that the nominal final state is a function of the maneuver parameters.

guess $\hat{\boldsymbol{p}}^0$, we seek the parameters that minimize a weighted two-norm of the final state deviation:

$$\hat{\boldsymbol{p}}^0 = \underset{\hat{\boldsymbol{p}}}{\text{argmin}} ||\hat{\boldsymbol{s}}_f^{nom}(\hat{\boldsymbol{p}}) - \hat{\boldsymbol{s}}_f^{des}||, \tag{30}$$

where $\hat{\boldsymbol{s}}_f^{des}$ denotes the desired final state after the maneuver. The solution to (30) is found by a numerical optimizer, where the final state function (29) is evaluated using numerical integration.[3] The initial covariance matrix of the maneuver parameters $\Sigma[\boldsymbol{p}^0]$ is a design parameter. Its entries are chosen to satisfy a trade-off between fast learning and outlier rejection.

B. Parameter Improvement Scheme

For the parameter improvement scheme, we numerically compute[4] the Jacobian matrix $\boldsymbol{J}$ of the nominal final state $\hat{\boldsymbol{s}}_f^{nom}(\hat{\boldsymbol{p}})$ with respect to the maneuver parameters $\hat{\boldsymbol{p}}$, around the initial parameter guess $\hat{\boldsymbol{p}}^0$:

$$\boldsymbol{J} = \left.\frac{\partial \hat{\boldsymbol{s}}_f^{nom}(\hat{\boldsymbol{p}})}{\partial \hat{\boldsymbol{p}}}\right|_{\hat{\boldsymbol{p}}=\hat{\boldsymbol{p}}^0}, \tag{31}$$

where $\hat{\boldsymbol{s}}_f^{nom}(\hat{\boldsymbol{p}})$ is given by (29). A crucial requirement of the presented parameter adaptation scheme is that the rank of the Jacobian matrix $\boldsymbol{J}$ is equal to or higher than the number of error states; otherwise it is in general not possible to correct for all elements of the observed state deviation after the open-loop maneuver (denoted as $\Delta \boldsymbol{s}_f$). If the number of adaptation parameters equals the number of states, the correction matrix $\boldsymbol{C}$ is defined to be the inverse of the Jacobian matrix $\boldsymbol{J}$:

$$\boldsymbol{C} := \boldsymbol{J}^{-1}. \tag{32}$$

On the other hand, if the number of adaptation parameters exceeds the number of error states, the correction matrix $\boldsymbol{C}$ may be defined, for example, as the least square solution:

$$\boldsymbol{C} := \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}. \tag{33}$$

For an observed final state offset $\Delta \boldsymbol{s}_f^i$ (the superscript i denotes the iteration number), the parameter adaptation rule is then defined as

$$\hat{\boldsymbol{p}}^{i+1} := \hat{\boldsymbol{p}}^i - \gamma^i \boldsymbol{C} \Delta \hat{\boldsymbol{s}}_f^i, \tag{34}$$

where $\gamma^i \in (0, 1]$ is the learning step size of the current iteration.[5] We choose γ^i such that the trace of the new parameter covariance matrix $\Sigma[\boldsymbol{p}^{i+1}]$ is minimized; this is known as the minimum mean square error estimate [15]. The optimal step size is given by

$$\gamma^i = \frac{\text{Trace}[\Sigma[\boldsymbol{p}^i]]}{\text{Trace}[\Sigma[\boldsymbol{p}^i]] + \text{Trace}[\boldsymbol{C}\Sigma[\Delta \boldsymbol{s}_f^i]\boldsymbol{C}^T]}, \tag{35}$$

and the covariance matrix of the new parameter set yields

$$\Sigma[\boldsymbol{p}^{i+1}] = (1-\gamma^i)^2 \Sigma[\boldsymbol{p}^i] + (\gamma^i)^2 \boldsymbol{C}\Sigma[\Delta \boldsymbol{s}_f^i]\boldsymbol{C}^T. \tag{36}$$

[3]The optimization is executed with *Matlab* [14], using the function *fminsearch* for the minimization of (30), and *ode45* for the integration of (29).

[4]Computations are executed with *Matlab* [14], using the function *jacobianest* of the toolset *Adaptive Robust Numerical Differentiation*.

[5]For $\gamma^i \in (0, 1]\ \forall i$, it can be shown that to first order the expected error converges to zero [9].

By inspection of (35) and (36) we find that the trace of the parameter covariance matrix $\Sigma[\boldsymbol{p}^i]$ always becomes smaller with increasing iteration number i, such that the learning step size γ^i, given by (35), converges to zero. If required, we can add an additional term $\Sigma[\tilde{\boldsymbol{p}}^i]$ to (36) that can be considered as process noise of the maneuver parameters $\boldsymbol{p}^i$. Doing so, the step size γ^i does not converge to zero, meaning that the learning algorithm is still adaptive after a large number of iterations.

After each parameter update, we also have to recompute the trajectories of the desired total thrust force f_{tot}^{des} and of the desired body rates $\boldsymbol{\omega}^{des}$, as these are the inputs fed to the on-board controller when the open-loop maneuver is executed.

C. Final State Offset Estimate

We now address the problem of estimating the final state offset $\Delta \boldsymbol{s}_f$ by observing one execution of the open-loop maneuver. Note that for reasons of readability the superscript i, denoting to current learning iteration, is omitted in the following. Fig. 3 shows the different intervals of such a learning iteration: The vehicle executes the high-performance open-loop maneuver between time t_0 and t_f, and recovers to hover between time t_f and t_r. During recovery, the vehicle receives control commands from an external pilot that can take advantage of absolute position measurements. We therefore assume that the vehicle is guided back to the position from where it started the open-loop maneuver.

1) Forward State Integration: We assume that the vehicle is hovering before the maneuver is triggered; the expected initial state $\hat{\boldsymbol{s}}_0$ is known, and the initial covariance matrix $\Sigma[\boldsymbol{s}_0]$ is a design parameter. (Roughly speaking, it is a measure for how precise the external pilot is able to hover.) By applying the estimator introduced in Section III throughout the open-loop maneuver (from time t_0 to t_f), we obtain a first estimate of the final state, denoted as $\boldsymbol{s}(t_f^-)$. The open-loop maneuver is assumed to be highly dynamic, hence

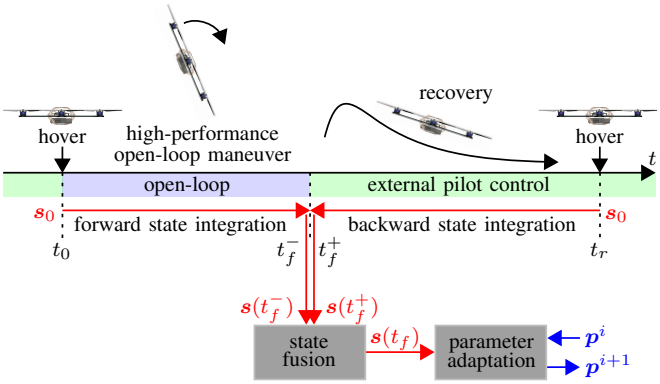

Fig. 3. Illustration of one learning iteration. On the timeline (top), the vehicle is shown performing the open-loop maneuver, followed by the recovery back to hover. Two estimates of the final state $\boldsymbol{s}(t_f)$ are computed for each iteration: one by forward integration from time t_0 to t_f, and one by backward integration from time t_r to t_f. Subsequently, the two estimates are fused and used to perform a parameter adaptation, reducing the expected value of the next final state deviation.

unmodeled aerodynamic effects might become significant. To account for this, the process noise terms $\tilde{\boldsymbol{a}}_{model}$ and $\tilde{\boldsymbol{\omega}}_{model}$ are increased as the maneuver is executed.

2) Backward State Integration: After recovery, at time t_r, the vehicle is assumed to be hovering again, i.e. $\hat{\boldsymbol{s}}(t_r) = \hat{\boldsymbol{s}}_0$ and $\Sigma[\boldsymbol{s}(t_r)] = \Sigma[\boldsymbol{s}_0]$ are known. Applying the state estimator backwards in time from t_r to t_f delivers a second estimate of the final state, denoted as $\boldsymbol{s}(t_f^+)$. The recovery performed by the external pilot is assumed to be less aggressive than the open-loop maneuver; the process noise is smaller. Consequently, the model-based estimates deliver valuable information leading to a more accurate final state observation.

3) State Estimate Fusion: Since the learning algorithm introduced above requires a single estimate of the final state $\boldsymbol{s}(t_f)$, the two estimates from forward and backward state integration are fused. Assuming that the two estimates are not correlated, the covariance matrix $\Sigma[\boldsymbol{s}(t_f)]$ of the fused estimate is given by

$$\Sigma[\boldsymbol{s}(t_f)] = (\Sigma[\boldsymbol{s}(t_f^-)]^{-1} + \Sigma[\boldsymbol{s}(t_f^+)]^{-1})^{-1}, \tag{37}$$

and the expected value $\hat{\boldsymbol{s}}(t_f)$ yields

$$\hat{\boldsymbol{s}}(t_f) = \Sigma[\boldsymbol{s}(t_f)](\Sigma[\boldsymbol{s}(t_f^-)]^{-1}\hat{\boldsymbol{s}}(t_f^-) + \Sigma[\boldsymbol{s}(t_f^+)]^{-1}\hat{\boldsymbol{s}}(t_f^+)). \tag{38}$$

Note that, for the two-dimensional case, the evaluation of (37) and (38) requires five-dimensional matrix inversions, which might be a burden for on-board implementation. Hence, scalar fusion of the state vector elements using the corresponding variances might be applied in cases where the limits of the computational resources are reached. (However, the matrix inversions must be computed only once for every learning iteration.) Finally, the estimated offset $\Delta\boldsymbol{s}_f$ from the desired state is obtained by

$$\Delta\hat{\boldsymbol{s}}_f = \hat{\boldsymbol{s}}(t_f) - \hat{\boldsymbol{s}}_f^{des}, \tag{39}$$

$$\Sigma[\Delta\boldsymbol{s}_f] = \Sigma[\boldsymbol{s}(t_f)], \tag{40}$$

which is then fed to the parameter adaptation scheme (34) and (36).

This completes the derivation of the iterative learning algorithm that is based on inertial sensor measurements and inputs from an external pilot during recovery.

V. Experimental Results

In this section, we provide experimental results of the state estimation and learning algorithm introduced above.

A. Experimental Platform

The experiments presented in the following are carried out in the Flying Machine Arena at ETH Zurich [16]. Modified Ascending Technologies 'Hummingbird' quadrocopters are used, where the electronics have been replaced by a Pixhawk PX4 Flight Management Unit (FMU) [17] and some other custom electronic components. The testbed provides an infrared motion-tracking system, which can be used for measuring ground truth and for simulating an external pilot.

B. Implementation

The estimator introduced in Section III and the learning strategy presented in Section IV are implemented on the PX4 FMU. The step size T of the estimator is chosen to be 1 ms. While the forward state integration runs in real-time, the backward integration can only be executed once the recovery is finished. Therefore, all sensor readings and commanded motor forces are written into a buffer during recovery, and then processed backwards in time as soon as the recovery is finished. (A constant recovery duration of 3 s is chosen.) The vehicle receives control commands from the ground station (i.e. from the external pilot) at a rate of 50 Hz. The external pilot is simulated by a position controller, which uses the position and attitude measurement data from the motion-capture system. The on-board controller, which tracks the desired body rates using rate gyro feedback, runs at 1000 Hz, i.e. at the same frequency as the estimator.

C. Parameterized Triple Flip Maneuver

The hover-to-hover triple flip is used as an example for a high-performance open-loop maneuver. Assuming that the maneuver starts at the origin, the desired final state is

$$\hat{\boldsymbol{s}}_f^{des} = (0, 0, 0, 0, 6\pi). \tag{41}$$

We choose a parameterization that is similar to the one presented in [9]: The maneuver is divided into seven intervals, as illustrated in Fig. 4, and each interval has constant propeller forces. The forces f_2 and f_4 are chosen to be the average of f_1 and f_3:

$$f_2 = f_4 = (f_1 + f_3)/2. \tag{42}$$

Consequently, each interval j is defined by the parameter triple $\{f_1^j, f_3^j, T_{int}^j\}$, where T_{int}^j denotes the duration of the interval. We choose the following parameter vector $\boldsymbol{p}$ to be adapted during the learning iterations:

$$\begin{aligned} p_1 &= f_1^1 - f_3^1, && \text{with } f_1^1 + f_3^1 = mg/2, \\ p_2 &= T_{int}^2, \\ p_3 &= T_{int}^4, \\ p_4 &= f_1^6 + f_3^6, && \text{with } f_1^6 = f_3^6, \\ p_5 &= f_3^7 - f_1^7, && \text{with } f_1^7 + f_3^7 = mg/2. \end{aligned} \tag{43}$$

These parameters are chosen because they allow reasonable adaptation authority for each element of the final state vector. Note that since we use a two-dimensional model, out-of-plane deviations are not eliminated. Due to the symmetry of the vehicle and the maneuver, however, the repeatable components of such disturbances are small.

D. Results

Fig. 5 shows the evolution of the estimated state offset versus the iteration number. For reference, the state offset measured with the motion-capture system is plotted as well, which in this context can be considered to be the ground truth. After about ten iterations the errors converge to an approximate value of zero, while we continue to see some non-repeatable disturbances in each iteration. Further, we can see that the position estimates are not very accurate, which is

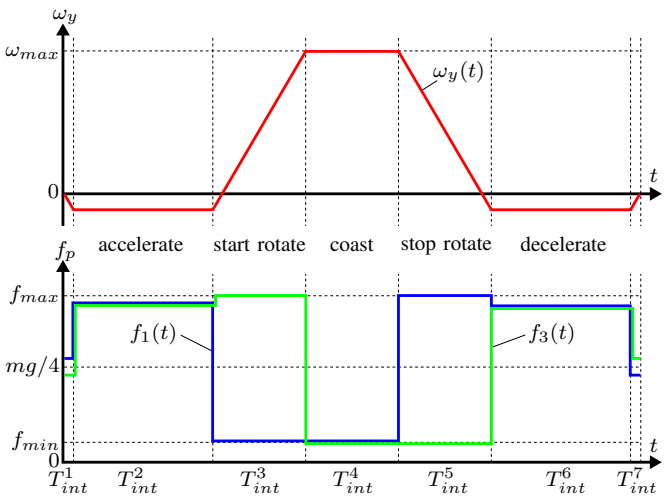

Fig. 4. Angular rate and propeller force trajectories of the parameterized triple flip maneuver. The maneuver consists of seven intervals, each defined by the forces f_1^j and f_3^j, and the interval duration T_{int}^j. The first interval is used to tilt a little bit, in order to end up at the right x-position at the end of the maneuver. During the second interval, the vehicle accelerates with almost full thrust in order to gain vertical momentum. The third interval is then used to gain rotational speed, applying the maximum rotational acceleration, and during the fourth interval the minimum thrust is applied, the vehicle just coasts. The intervals five to seven are then the inverse of the intervals one to three, i.e. stop rotating, decelerate, and adjust tilt angle.

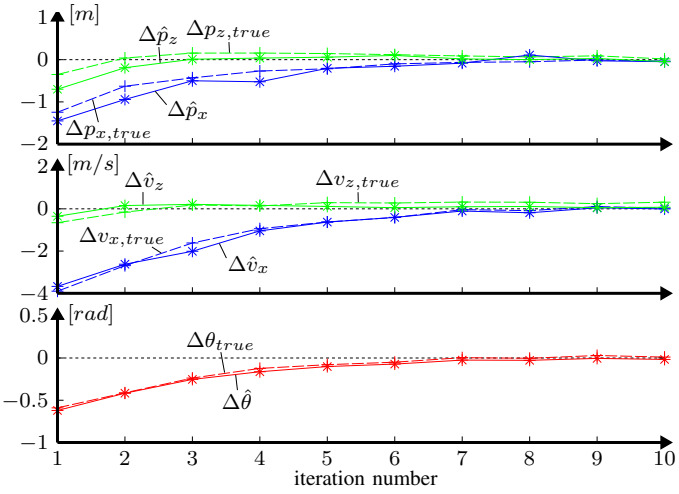

Fig. 5. Final state offset versus iteration number for position (top), velocity (middle), and pitch angle (bottom). The solid lines show the estimated values obtained by the estimator introduced in Section III. The dotted lines show the true values measured by a motion-capture system. After ten learning iterations, all errors have converged to a value near zero.

expected to some extent because of the integrator drift. Even if the estimate deviates from the true value quantitatively, however, it delivers useful qualitative information for the learning algorithm: The direction of the estimated deviation is correct for most iterations, especially for low iteration numbers. The state estimate thus delivers valuable information about the direction towards which the maneuver parameters must be adapted. Hence, while the estimator introduced in Section III might not be accurate enough to be fed to a feedback controller for example, in combination with the learning algorithm presented herein it delivers useful results when improving the performance of a triple flip maneuver.

VI. Conclusion

We have presented a learning scheme for high-performance open-loop maneuvers with quadrocopters, which can be implemented on-board and requires no external motion-capture system. The method has been validated by reducing the systematic errors of a triple flip maneuver.

For the experiments presented herein the external pilot was simulated by a position controller using motion-capture system data; the recovery was executed in a fast and repeatable fashion, which might not be the case for a human pilot. Hence, this paper shows that the method works in principle, but whether it is applicable with a human pilot is still an open question and part of future work. Another future subject of study is the experimental validation on a three-dimensional maneuver; while herein the presented approach has been applied to a two-dimensional example, in principle the method should be applicable to the three-dimensional case, too.

References

[1] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *Proceedings of the International Symposium on Experimental Robotics*, 2010.

[2] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011.

[3] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012.

[4] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of robotic systems*, vol. 1, no. 2, 1984.

[5] G. Casalino and G. Bartolini, "A learning procedure for the control of movements of robotic manipulators," in *IASTED symposium on robotics and automation*, 1984.

[6] J. J. Craig, *Adaptive control of manipulators through repeated trials*. General Motors Research Laboratories, 1983.

[7] A. Tayebi, "Adaptive iterative learning control for robot manipulators," *Automatica*, vol. 40, no. 7, 2004.

[8] A. P. Schoellig, F. L. Mueller, and R. DAndrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33, no. 1-2, 2012.

[9] S. Lupashin and R. D'Andrea, "Adaptive fast open-loop maneuvers for quadrocopters," *Autonomous Robots*, vol. 33, no. 1-2, 2012.

[10] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, 2012.

[11] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012.

[12] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[13] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, 2004.

[14] The MathWorks Inc., "Matlab, Version R2012a (7.14.0.739)," 2012.

[15] B. D. Anderson and J. B. Moore, *Optimal filtering*. DoverPublications. com, 2012.

[16] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, vol. 24, no. 1, 2014.

[17] "PX4 FMU," www.pixhawk.ethz.ch/px4/modules/px4fmu, accessed 2013-09-11.