Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight

Matthias Faessler, Davide Falanga, and Davide Scaramuzza

Abstract—**Quadrotors are well suited for executing fast maneuvers with high accelerations but they are still unable to follow a fast trajectory with centimeter accuracy *without* iteratively *learning* it beforehand. In this paper, we present a novel body-rate controller and an *iterative* thrust-mixing scheme, which improve the trajectory-tracking performance without requiring learning and reduce the yaw control error of a quadrotor, respectively. Furthermore, to the best of our knowledge, we present the first algorithm to cope with motor saturations smartly by *prioritizing* control inputs which are relevant for stabilization and trajectory tracking. The presented body-rate controller uses LQR-control methods to consider both the body rate and the single motor dynamics, which reduces the overall trajectory-tracking error while still rejecting external disturbances well. Our iterative thrust-mixing scheme computes the four rotor thrusts given the inputs from a position-control pipeline. Through the iterative computation, we are able to consider a varying ratio of thrust and drag torque of a single propeller over its input range, which allows applying the desired yaw torque more precisely and hence reduces the yaw-control error. Our prioritizing motor-saturation scheme improves stability and robustness of a quadrotor's flight and may prevent unstable behavior in case of motor saturations. We demonstrate the improved trajectory tracking, yaw-control, and robustness in case of motor saturations in real-world experiments with a quadrotor.**

Supplementary Material

A video showing the conducted experiments with a quadrotor is available at: https://youtu.be/6YEMxFgToyg

I. Introduction

A. Motivation

In recent years, autonomous quadrotors became very popular due to their agility, allowing them to execute aggressive maneuvers. We consider a trajectory to be aggressive if at least one of the quadrotor's motors gets close to saturation during its execution, which is the case if large linear or angular accelerations are required. Even today, when executing such an aggressive trajectory with a quadrotor, the tracking errors without replanning or iteratively learning the maneuver beforehand can be large. For state of the art quadrotor control methods, trajectory tracking errors of up to several body lengths during execution of a fast trajectory (without learning) are reported in e.g. [1] and [2]. Such errors are too large for e.g. fast obstacle avoidance in cluttered environments where iterative learning cannot be applied due to non repetitive motions.

At the heart of precise trajectory tracking with a quadrotor is a body-rate controller that tracks the desired body rates which are typically computed from a high-level control pipeline (e.g. [3], [4]). Only when tracking the desired body rates well, the quadrotor can apply its desired attitude and with that the desired thrust direction precisely, which in turn enables precise translations. To achieve good body-rate tracking, it is crucial to apply the single rotor thrusts precisely, such that the desired body torques and collective thrust can be applied correctly.

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—http://rpg.ifi.uzh.ch.

This work was supported by the National Centre of Competence in Research Robotics (NCCR) and the DARPA FLA Program.

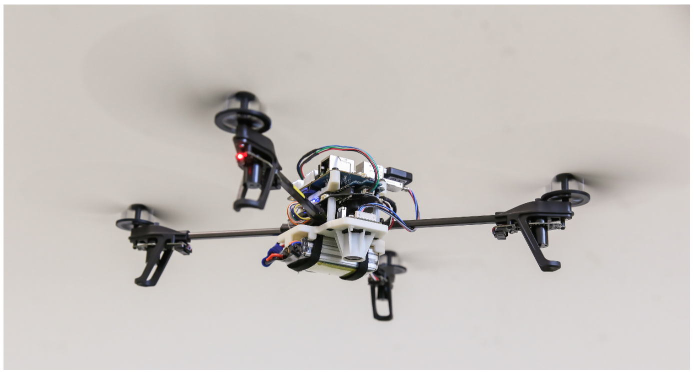

Fig. 1: Our quadrotor used for the experiments of this work.

To fully exploit the agility of quadrotors, it is desirable to design aggressive trajectories. However, during trajectory design, it is difficult to ensure feasibility while trying to exploit the entire range of feasible motor inputs. And even a trajectory that is feasible with respect to motor saturations cannot guarantee that the tracking controller does not compute motor inputs exceeding its limits due to deviations from the reference trajectory. If such a saturation of one or several motors occurs, the quadrotor may deviate substantially from its reference trajectory or even get unstable if it is not handled correctly.

B. Contribution

The contribution of this work is threefold. First, we design a novel body-rate controller using LQR methods, which takes the dynamics of the single motors with propellers into account. Compared to previously applied controllers, it improves trajectory tracking while maintaining the same disturbance-rejection performance. Second, we improve the computation of the single rotor thrusts such that the desired yaw torque on the quadrotor body, given by a feedback controller, is reached more precisely than with state-of-the-art methods. To do so, we consider that the ratio of thrust and drag torque of a single propeller is not constant over the entire input range, as it is instead assumed in the literature ([3], [4], [5]). Third, we increase the quadrotor's stability and its robustness in case of motor-input saturations. We tackle this by applying a saturation scheme for the single rotor thrusts, which prioritizes between the desired collective thrust and body torques according to their relevance for stabilizing the quadrotor and following a trajectory.

C. Related Work

Many state-of-the-art quadrotor-control pipelines are making use of a two-level architecture with a high-level position controller and a proportional low-level body-rate controller ([3], [4], [6]), which we also consider in this paper. Besides such cascaded loops of proportional controllers, LQR [7] and nonlinear model predictive control techniques on SO(3) [8] were successfully applied to control the full attitude of quadrotors. In [9], cascaded PID controllers are designed and enhanced with Smith predictors to incorporate the dynamics of the motors for full quadrotor attitude control on SO(3). An LQR attitude controller for a single axis, which is extended with first order dynamics of the motors is presented in [10] and [11]. In contrast, we design an LQR controller for the coupled 3D body rates, incorporating the motor dynamics, which also provides feedback linearization and feed forward on desired angular accelerations.

In both [3] and [4], the low-level control part outputs a desired collective thrust and body torques that need to be applied to the quadrotor's body by the four single rotor thrusts. In other state-of-the-art quadrotor controllers [12], [13], the high-level part directly outputs the desired collective thrust and body torques. In all these works, the desired collective thrust and body torques need to be converted into four single rotor thrusts which can then be applied by knowing the mapping from motor commands to rotor thrusts, denoted as *thrust mapping*. This is commonly done by solving a system of four equations for the four rotor thrusts, assuming that the ratio of thrust and drag torque of a rotor is constant over its entire motor-input spectrum. We refer to the process of computing the four single rotor thrusts as *thrust mixing*. In this paper, we propose an iterative thrust-mixing scheme that does not require the assumption that the ratio of thrust and drag torque of a rotor is constant, which then allows applying the desired yaw torque more accurately.

Commonly, polynomial trajectories are designed for quadrotors since they are easy to handle mathematically and are dynamically feasible if they are continuous up to a sufficient order of derivatives. They can be generated with a global optimization in which they can be constrained at the start, end, and intermediate waypoints ([13], [14]). Between the waypoints, their feasibility with respect to input limitations cannot be guaranteed. Methods for checking the feasibility of the entire trajectory after generation are proposed in [15] and [16] by simplifying the problem with conservative approximations which do not allow using the full range of feasible inputs. Also, during trajectory execution, it is not guaranteed that the controller does not compute any infeasible inputs since the quadrotor may deviate from a feasible trajectory. Model predictive control methods [17] are able to incorporate input feasibility constraints but are computationally not suitable a low-level controllers running on a micro controller while it is also difficult for them to make use of the entire available

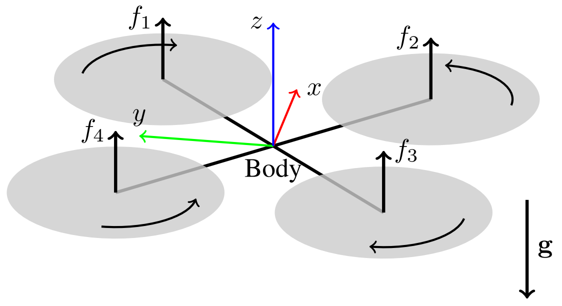

Fig. 2: Quadrotor with coordinate system and single rotor forces.

range of inputs. Instead of considering feasibility constraints during trajectory generation, in [18] a partial control allocation method that prioritizes the application of desired body torques over collective thrust is used to handle infeasible inputs before applying the motor commands. In this work, we present a saturation scheme that prioritizes control inputs according to their importance for trajectory tracking in case of motor saturations. The presented saturation scheme is able to make use of the full input range of the individual motors.

II. System Overview

We consider a quadrotor that is modeled as a rigid body which is controlled by four single rotor thrusts f_i as illustrated in Fig. 2. By changing these four single rotor thrusts, a three axis torque $\boldsymbol{\eta}$ and a mass normalized collective thrust c can be applied on the quadrotor's body. The relation of the single rotor thrusts to the collective thrust and the body torques can be formulated using the coordinate system of Fig. 2 as

$$\boldsymbol{\eta} = \begin{bmatrix} \frac{\sqrt{2}}{2} l(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2} l(-f_1 - f_2 + f_3 + f_4) \\ \kappa_1 f_1 - \kappa_2 f_2 + \kappa_3 f_3 - \kappa_4 f_4 \end{bmatrix}, \tag{1}$$

$$mc = f_1 + f_2 + f_3 + f_4, \tag{2}$$

where l is the quadrotor's arm length, $\kappa_i = \kappa(f_i)$ is a coefficient relating the drag torque and the thrust of a single rotor, and m is the quadrotor's mass. Note that unlike in [4] and [6], we consider the rotor drag torque coefficient κ to be a function of the rotor thrust (cf. Fig 4) and *not a constant*.

We model the single rotor thrust f and drag torque τ as quadratic polynomials of the motor input u as

$$f(u) = k_2^f u^2 + k_1^f u + k_0^f, \tag{3}$$

$$\tau(u) = k_2^\tau u^2 + k_1^\tau u + k_0^\tau, \tag{4}$$

where the coefficients k_j^f and k_j^τ are identified by running a single motor with a propeller on a load cell and measuring the resulting forces and moments. The motor input u corresponds to the command we can send to our electronic speed controllers in the range $[-1,\ 1]$. We chose to have three coefficients since it approximates the measured values better than modeling the rotor thrust and drag torque with only a quadratic term, as proposed in e.g. [5]. Figure 3 compares the two methods for fitting the thrust mapping. From (3) and (4),

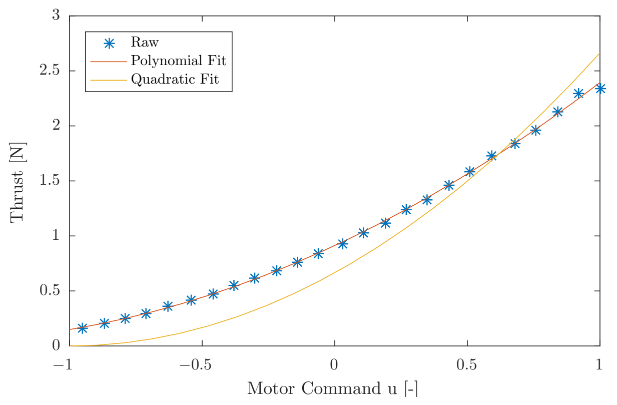

Fig. 3: To approximate the thrust mapping for a single motor with propeller, we fit a second order polynomial into raw thrust measurements obtained by running the motor on a load cell. The polynomial fit approximates the measurements much better than a purely quadratic fit of the form $f(u) = k_2^f u^2$.

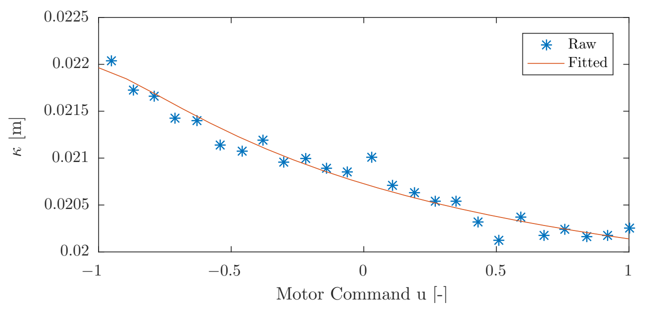

Fig. 4: Values for κ estimated from load cell data obtained by dividing drag torque and thrust values for a motor and propeller as described in Section VI-A. The fitted curve is the ratio of the fitted quadratic functions for measured thrust and drag torque in (3) and (4). The range of motor commands is normalized to [-1, 1].

we can compute the rotor drag torque coefficient as

$$\kappa(f) = \frac{\tau\left(u = \frac{-k_1^f + \sqrt{\left(k_1^f\right)^2 - 4k_2^f\left(k_0^f - f\right)}}{2k_2^f}\right)}{f}. \tag{5}$$

Figure 4 shows the identified values for the rotor drag torque coefficient and how it varies by about 10 % over the entire range of available motor inputs.

As in [4] and [6], we consider that the thrust mapping (3) can be refined with rotor-fitness factors γ_i that relate the true thrust f_i and the thrust identified with a load cell $\check{f}_i$ for a certain motor input as $f_i = \gamma_i \check{f}_i$. The rotor fitness factors can be estimated by averaging the applied rotor thrust commands during hover flight.

III. Body Rate Control

Due to a hardware architecture with two processing units for high-level and low-level control on our quadrotors, we split the controllers such that the high-level controller computes desired body rates and the low-level controller tracks them. In this section, we present a novel body-rate controller that improves the body-rate-tracking performance by also considering the dynamics of the motors. We achieve this by designing an LQR controller for a dynamical system containing the body rates and body torques as state. The inputs to this controller are the desired body rates $\boldsymbol{\omega}_{des}$ and the desired mass normalized collective thrust c_{des}, which we assume are given from a high-level position controller (e.g. from [3]).

The dynamics of the quadrotor's body rates $\boldsymbol{\omega}$ are

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1} \cdot (\boldsymbol{\eta} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}), \tag{6}$$

where $\boldsymbol{J}$ is the moment of inertia of the quadrotor. Additionally, we model the dynamics of the single rotor thrusts as first order systems

$$\dot{f} = \begin{cases} \frac{1}{\alpha_{up}}(f_{des} - f) & \text{if } f_{des} \geq f \\ \frac{1}{\alpha_{dwn}}(f_{des} - f) & \text{if } f_{des} < f \end{cases}, \tag{7}$$

where the time constants α_{up} and α_{dwn} are identified from applying step inputs to a single motor with propeller on a load cell. From these load-cell experiments we found that the single rotor thrust dynamics are considerably different when spinning a motor up or down. Now we can use (6) and (7) to establish a dynamical system with state $\mathbf{s} = [\boldsymbol{\omega}^T \; \boldsymbol{\eta}^T]^T$ and input $\mathbf{u} = \boldsymbol{\eta}_{des}$. To create a simplified model for the dynamics of the body torques from (1) and (7), we approximate $\alpha_{up} = \alpha_{dwn} = \alpha$. In practice, we found $\alpha = (\alpha_{up} + \alpha_{dwn})/2$ to be a good approximation which stems from the fact that for changing a body torque, we make use of opposite rotors where one spins up and the other one down. Additionally, we simplify the dynamics of $\boldsymbol{\eta}_z$ by approximating κ to be constant and not depend on the rotor thrust, which leads to the following dynamics of the body torques:

$$\dot{\boldsymbol{\eta}} = \frac{1}{\alpha}(\boldsymbol{\eta}_{des} - \boldsymbol{\eta}). \tag{8}$$

Note that the introduced simplifications of these dynamics are necessary for the following feedback-controller design. Linearizing (6) and (8) around $\boldsymbol{\omega} = \mathbf{0}$ and $\boldsymbol{\eta} = \mathbf{0}$ leads to the system

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \boldsymbol{J}^{-1} \\ \mathbf{0} & -\frac{1}{\alpha}\mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\eta} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \frac{1}{\alpha}\mathbf{I}_3 \end{bmatrix} \boldsymbol{\eta}_{des}, \tag{9}$$

which we can use to design an infinite-horizon LQR control law $\mathbf{u} = -\mathbf{K}_{lqr}\mathbf{s}$ that minimizes the cost function

$$\int \mathbf{s}^T \mathbf{Q} \mathbf{s} + \mathbf{u}^T \mathbf{R} \mathbf{u} \, dt, \tag{10}$$

where $\mathbf{Q}$ is a diagonal weight matrix and $\mathbf{R}$ is the identity matrix. The solution to the formulated LQR problem is a gain matrix of the form

$$\mathbf{K}_{lqr} = \begin{bmatrix} k_{\omega_{xy}} & 0 & 0 & k_{\eta_{xy}} & 0 & 0 \\ 0 & k_{\omega_{xy}} & 0 & 0 & k_{\eta_{xy}} & 0 \\ 0 & 0 & k_{\omega_z} & 0 & 0 & k_{\eta_z} \end{bmatrix}, \tag{11}$$

which corresponds to a PD controller of the body rates. Additionally, we add feed forward terms such that $\boldsymbol{\omega}_{des}$ is reached with $\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}}_{des}$, resulting in the control policy

$$\boldsymbol{\eta}_{des} = \mathbf{K}_{lqr} \begin{bmatrix} \boldsymbol{\omega}_{des} - \hat{\boldsymbol{\omega}} \\ \boldsymbol{\eta}_{ref} - \hat{\boldsymbol{\eta}} \end{bmatrix} + \hat{\boldsymbol{\omega}} \times \boldsymbol{J}\hat{\boldsymbol{\omega}} + \boldsymbol{J}\dot{\boldsymbol{\omega}}_{des}, \tag{12}$$

with $\boldsymbol{\eta}_{ref} = \boldsymbol{\omega}_{des} \times \boldsymbol{J}\boldsymbol{\omega}_{des} + \boldsymbol{J}\dot{\boldsymbol{\omega}}_{des}$ computed from (6). The vector $\hat{\boldsymbol{\omega}}$ are the estimated body rates measured by the onboard gyroscopes, and $\hat{\boldsymbol{\eta}}$ are the estimated body torques obtained by estimating the single rotor thrusts with (7) and using (1) to transform them into body torques. Note that this estimation makes use of the non-symmetric model (7) of the rotor thrusts, i.e., $\alpha_{up} \neq \alpha_{dwn}$, while for the control design we assumed $\alpha_{up} = \alpha_{dwn}$. Also note that this estimation can be improved if feedback of the rotor speeds is available. In the controller (12), the term $\hat{\boldsymbol{\omega}} \times \boldsymbol{J}\hat{\boldsymbol{\omega}}$ provides feedback linearization, compensating for the coupling terms in the body-rate dynamics, and the term $\boldsymbol{J}\dot{\boldsymbol{\omega}}_{des}$ can be used as feed forward on desired angular accelerations that can be computed from a trajectory to be tracked due to the quadrotor's differential flatness property [13].

IV. Iterative Thrust Mixing

To compute the single rotor thrusts that achieve the desired body torques $\boldsymbol{\eta}_{des}$ and collective thrust c_{des}, which we denote as *mixer inputs*, we have to solve (1) and (2) for f_i. Since we consider the rotor drag torque coefficients to be a function of the rotor thrust, we cannot solve this system of equations directly, but we can do so iteratively.

To initialize the iteration, we start by setting the single rotor thrusts equal such that they achieve the desired collective thrust:

$$f_i = \frac{m c_{des}}{4}. \tag{13}$$

Note that these values are only used to compute the rotor drag torque coefficients in the first iteration. Then, we start the iteration with the following two steps: i) solve (5) to get κ_i, and ii) solve (1) and (2) with $\boldsymbol{\eta}_{des}$ and c_{des} for f_i. Additionally to quadrotors, this iterative scheme can also be applied to other multirotors.

V. Saturation with Input Priorities

Once we have computed the desired single rotor thrusts, we have to make sure that they lie within the feasible range $[f_{min}, f_{max}]$ for each single motor. Naively, this can be done by clipping each rotor thrust if its desired value is outside this range. This is a simple and fast procedure with the drawback that none of the desired mixer inputs $\boldsymbol{\eta}_{des}$ and c_{des} is achieved exactly if one of the rotor thrusts is clipped. Nonetheless, not all these mixer inputs are equally important in terms of the quadrotor's ability to stabilize and track a trajectory. Since a quadrotor can only produce a collective thrust in its body upwards direction, it has to be aligned with the desired acceleration for following a trajectory in 3D space. The rotation around the thrust direction is irrelevant for the translational motion of the quadrotor. Therefore, we want to give least priority to achieve the desired yaw torque in case of an input saturation. On the other hand, the quadrotor uses roll and pitch torques to change its thrust direction which enables stabilization and therefore makes them the most important inputs. Furthermore, state of the art control methods for quadrotors (e.g. [3], [4]) are based on the assumption that the orientation of the thrust vector can be changed quickly. For these reasons, in case of an input saturation, we want to give highest priority to applying the desired roll and pitch torques, second highest priority to applying the desired collective thrust, and lowest priority to applying the desired yaw torque.

Algorithm 1 Rotor Thrust Saturation

```
Compute $f_i$ as detailed in Section IV
**Perform yaw-torque saturation:**
**if** Motor saturated **AND** $|\eta_{z,des}| > \eta_{z,assured}$ **then**
    Find rotor $j$ that violates thrust limits the most
    $f_j \leftarrow \gamma_j \cdot f_{limit}$
    Solve (1) and (2) for $f_{i \setminus j}$ and $\eta_z$
    **if** $\mathrm{sign}(\eta_{z,des}) \cdot \eta_z < \eta_{z,assured}$ **then**
        $\eta_z \leftarrow \mathrm{sign}(\eta_{z,des}) \cdot \eta_{z,assured}$
        Solve (1) and (2) for $f_i$
    **end if**
**end if**
**Perform collective-thrust saturation:**
**if** Motor saturated **then**
    **if NOT**(upper **AND** lower saturation reached) **then**
        Find rotor $j$ that violates thrust limits the most
        Shift $f_i$ equally s.t. $f_j = \gamma_j \cdot f_{limit}$
    **end if**
**end if**
Enforce single rotor thrust limits by thrust clipping
```

A. Yaw-Torque Saturation

We achieve this prioritization by a saturation scheme as summarized in Algorithm 1. First, the single rotor thrusts are computed according to Section IV. If one of the single rotor thrusts exceeds its limits, we try to change the applied yaw torque to avoid saturation, given that the desired yaw torque is above a certain minimum $\eta_{z,assured}$, which we can optionally impose. Such an assured yaw torque might be desired for applications where we want to guarantee that we always have some control on the heading of a quadrotor. In case of saturation, we do not apply the iterative mixer in order to save time since we are unable to apply the desired yaw torque anyways. To do the yaw-torque saturation, we find the rotor that violates the input limit the most, set it to the corresponding limit and then solve (1) and (2) for the remaining rotor thrusts and the yaw torque. If the resulting yaw torque is still above the value we want to assure, we successfully enforced all the rotor-thrust limits by only changing the applied yaw torque. In other words, in this case, Algorithm 1 guarantees that the quadrotor applies the desired roll and pitch torques and the desired collective thrust, but *not* the desired yaw torque. If the resulting yaw torque is below the value we want to assure, we set it to the assured value $\eta_{z,assured}$ and recompute the rotor thrusts.

B. Collective-Thrust Saturation

If one of the rotor-thrust limits is still violated, we try to change the applied collective thrust to avoid saturation. This is only possible if two rotors do not violate the upper and the lower limit simultaneously, in which case it is impossible to achieve the desired roll and pitch torques by changing the applied collective thrust. If only one limit is violated, we find the rotor that violates the input limit the most, set it to its limit and shift the remaining rotor thrusts by the same amount. In this case, Algorithm 1 ensures that the quadrotor applies the desired roll and pitch torques but *not* the desired collective thrust and *not* the desired yaw torque.

C. Thrust Clipping

At this point, if a rotor still violates its input limits, we have to apply thrust clipping and can therefore not achieve any of the desired mixer inputs precisely. Note that the presented procedure uses the full range of available thrusts for each rotor also considering individual rotor fitness factors.

VI. Experiments

A. Experimental Setup

We built our quadrotor from selected off-the-shelf components and custom 3D printed parts (see Fig. 1) with a total weight of $503\,\mathrm{g}$. It is based on the frame of the Parrot AR.Drone 2.0 including their motors, motor controllers, gears, and propellers. On this frame, we use a PX4IOAR adapter board and a PX4FMU autopilot that runs all the presented algorithms of this paper. All the details about our drone can be found in [6].

We identified the thrust mapping (3) and the torque mapping (4) by putting one single motor with propeller on a *ATI Mini40* load cell. The first order time constants $\alpha_{up} = 11\,\mathrm{ms}$ and $\alpha_{dwn} = 27\,\mathrm{ms}$ were estimated by applying step inputs to a motor on the load cell.

The following three subsections provide results of experiments that show the effects of applying our LQR body-rate controller, iterative mixer, and prioritizing saturation, respectively, as isolated as possible. All the flight experiments were conducted in an OptiTrack motion capture system to acquire a ground truth state measurement of the quadrotor. A video of the conducted experiments can be found on https://youtu.be/6YEMxFgToyg.

B. Body-Rate Controller

We compare the proposed LQR body-rate controller to a proportional controller from our previous work [3] (identical to [4]) in its disturbance rejection and trajectory-tracking performance. Our goal for the LQR controller is to maintain the disturbance-rejection performance of the previously used proportional controller but achieve better trajectory tracking. Note that when neglecting the motor dynamics in the LQR design, we obtain a proportional controller and hence compare a similar controller which either considers motor dynamics or not. We conduct all experiments for a low gain proportional controller (a), as used in [3], a high gain proportional controller (b), and the proposed LQR controller (c) with a proportional gain corresponding to the one of (b).

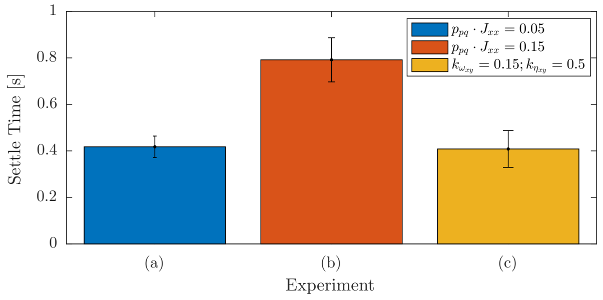

Fig. 5: Disturbance-rejection performance measured by the time it takes the quadrotor to settle after disturbing it by hitting one of the arms upwards (averaged over 10 disturbances). The experiments (a) and (b) are conducted with different gains using the body-rate controller from [3] or [4] and (c) is conducted with the proposed LQR controller. The vertical black lines indicate the standard deviation of the settling time.

We measure the disturbance-rejection performance by hitting one of the quadrotor's arms upwards during hover flight and measuring the time it takes it to settle again.[1] This is illustrated in Fig. 5, where the low gain proportional and the LQR controller have almost identical settling times and the high gain proportional controller has a significantly higher one. We measure the trajectory-tracking performance by flying a quadrotor back and forth in the x-direction with a maximum acceleration of $12\,\mathrm{m\,s^{-2}}$ and a maximum velocity of $5.7\,\mathrm{m\,s^{-1}}$ on a trajectory composed of multiple polynomial segments[1] that are continuous in snap (cf. Fig. 6) and measuring the tracking errors illustrated in Fig. 7. It shows that increasing the gain of the proportional controller improves trajectory tracking significantly. Our proposed LQR controller shows a slightly larger position error than the high gain proportional controller but still reduces it noticeably by more than $25\,\%$ compared to the low gain proportional controller. To provide a fair comparison of the two controllers, we apply the proposed iterative mixer and prioritizing saturation (only shortly active) in all the experiments.

Table I summarizes the results of the disturbance rejection and trajectory-tracking experiments to compare the three different controllers. It especially shows that increasing the gain of the proportional controller improves trajectory tracking but also reduces its disturbance-rejection performance significantly [from (a) to (c)]. On the other hand, the LQR controller (c) with equivalent gains to (b) can almost improve trajectory tracking as much but maintains the same disturbance-rejection performance of the low gain proportional controller (a).

C. Iterative Mixer

Figure 8 compares the performance of the proposed iterative mixer and a one-shot mixer in a three minute hover flight. The corresponding statistics in Table II show that the iterative mixer reduces the maximum, mean, and standard deviation of the yaw error by more than $35\,\%$. In this comparison, both methods make use of the same polynomial model for the thrust and torque mapping (3), (4), respectively. We conducted this

[1]See video on: https://youtu.be/6YEMxFgToyg

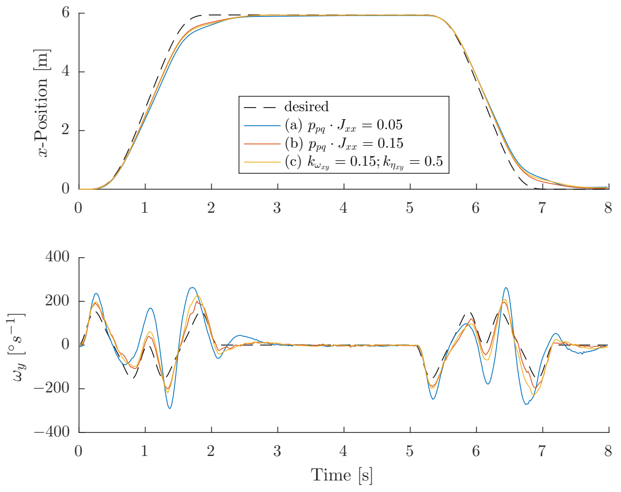

Fig. 6: Position and body-rate-tracking performance for a back-and-forth motion in the x-direction. The trajectory is composed of multiple polynomial segments with maximum acceleration of $12\,\mathrm{m\,s^{-2}}$ and maximum velocity of $5.7\,\mathrm{m\,s^{-1}}$. The experiments (a) and (b) are conducted with different gains using the body-rate controller from [3] or [4] and (c) is conducted with the proposed LQR controller. The corresponding errors are illustrated in Fig. 7.

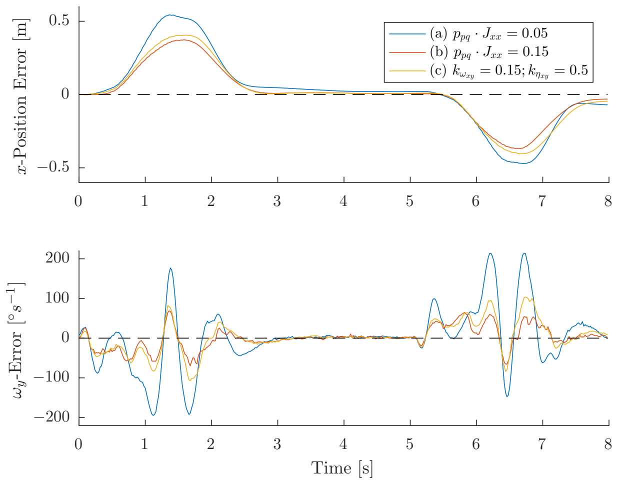

Fig. 7: Tracking errors for the trajectory in Fig. 6. The corresponding error statistics are presented in Table I, where we also compare the trajectory tracking and the disturbance-rejection performance of each controller.

experiment in hover flight to suppress non modeled dynamics and motor saturations in order to isolate the effect of the iterative mixer. Note that in non-hover flight, where larger roll and pitch torques are required, the iterative mixer becomes more advantageous due to larger commanded rotor-thrust differences. In practice, after the third mixer iteration, the error between the desired and achieved yaw torque becomes negligible ($\ll 1\,\%$) for different tested motor types. The error made by a one-shot mixer can be more than $5\,\%$ of the desired yaw torque. Note that this error depends on the differences of the commanded single rotor thrusts and vanishes when the four commanded thrusts are equal.

TABLE I: Position error in x-direction (e_x), thrust-direction error angle (θ), and pitch-rate error (e_{ω_y}) statistics for the experiments illustrated in Fig. 6 and averaged over twelve back-and-forth motions, as well as settle time (t_{settle}) statistics for the experiment illustrated in Fig. 5.

	(a)	(b)	(c)
$\mu(e_x)$ [cm]	**14.50**	**9.39**	**10.35**
$\sigma(e_x)$ [cm]	23.05	15.60	17.19
$\mu(\theta)$ [°]	5.55	3.84	4.14
$\sigma(\theta)$ [°]	7.33	5.00	5.22
$\mu(e_{\omega_y})$ [$^\circ\,\mathrm{s}^{-1}$]	42.28	14.88	21.52
$\sigma(e_{\omega_y})$ [$^\circ\,\mathrm{s}^{-1}$]	72.38	23.65	32.41
$\mu(t_{settle})$ [s]	**0.42**	**0.79**	**0.41**
$\sigma(t_{settle})$ [s]	0.05	0.09	0.08

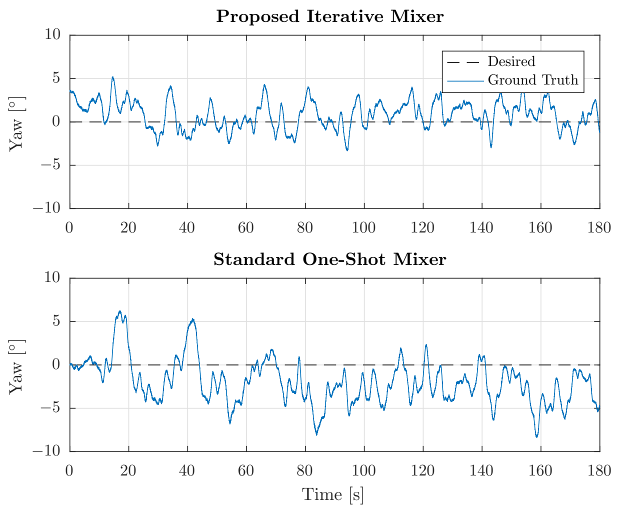

Fig. 8: Comparison of the actual and desired yaw angle during hover flight using the proposed iterative mixer (top) and a one-shot mixer (bottom). Statistics of the yaw error are depicted in Table II.

D. Prioritizing Saturation

The performance of the prioritizing saturation compared to thrust clipping is shown in Fig. 9 with an experiment[1] where a quadrotor is commanded to do a simultaneous step in height by $1\,\mathrm{m}$ and yaw by $90°$. In this experiment, the minimum and maximum single rotor thrust were artificially set to $0.8\,\mathrm{N}$ and $1.6\,\mathrm{N}$ respectively (hover thrust: $1.25\,\mathrm{N}$) to force the saturation scheme to become active without requiring an aggressive maneuver. It can be seen that with the prioritizing saturation, the x and y position stay close to their constant desired values, whereas they deflect a lot from the desired value in case of

TABLE II: Yaw error statistics comparison between a standard one-shot mixer and the proposed iterative mixer in hover flight.

	One-Shot	**Iterative**	
Max Yaw Error	8.371	**5.209**	[°]
Mean Yaw Error	2.667	**1.462**	[°]
Yaw Error Standard Deviation	2.549	**1.593**	[°]

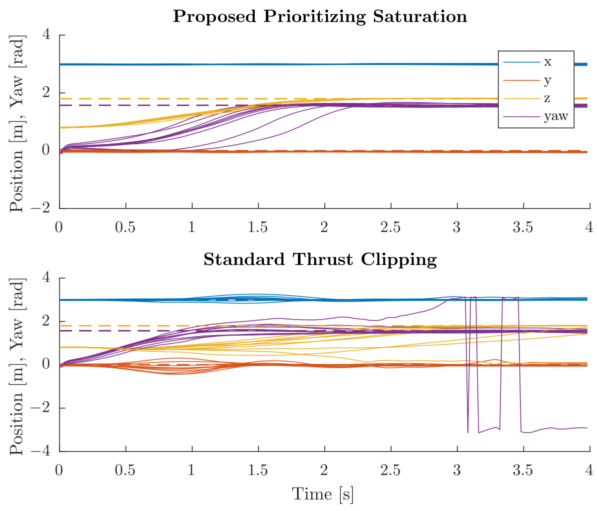

Fig. 9: Response in position and yaw for ten runs where the quadrotor simultaneously performs a 1 m step in height and a 90° step in yaw starting at $t = 0$ s using prioritizing saturation (top) and thrust clipping (bottom). The reference values after the step are: $x = 3.0$ m, $y = 0.0$ m, $z = 1.8$ m, and $yaw = \pi/2$ rad, and the assured yaw torque is $\eta_{z,assured} = 0.01$ N m. Solid lines show ground truth data and dashed lines show desired reference values.

TABLE III: Iterative mixing and saturation computation times. As comparison, the thrust mixing with a one-shot mixer takes 1.4 µs with 0.3 µs standard deviation.

	Mean [µs]	Standard Deviation [µs]
Thrust Mixing (3 iter.)	243.7	5.8
Yaw Saturation	166.6	4.1
Thrust Saturation	1.3	0.3
Thrust Clipping	1.5	0.4
Total	413.1	7.1

thrust clipping. This even causes the quadrotor to crash into the ground for one run when thrust clipping is applied. On the other hand, with the prioritizing saturation, compared to thrust clipping, it takes longer for the yaw angle to settle on the desired value since its priority is the smallest. Note that when a saturation occurs, we do not make use of the iterative mixer since we are unable to reach the desired yaw torque due to the saturation. The computation times of the different steps in the presented saturation scheme are summarized in Table III.

VII. Conclusion

We presented an LQR based body-rate controller, an iterative mixer to compute the desired single rotor thrusts, and a prioritizing motor-saturation scheme which we all evaluated in real experiments with a quadrotor in a motion capture system. Compared to the state-of-the-art proportional controllers, the proposed body-rate controller almost halved the body-rate tracking error and reduced the position error by more than 25 % during a fast trajectory while preserving its damping properties against external disturbances. The iterative mixer reduces the yaw error by considering a non constant ratio of thrust and drag torque of a single rotor by more than 35 % in hover flight. The presented motor saturation scheme prioritizes roll and pitch, which are most important for stabilization and trajectory tracking, before collective thrust and yaw. We demonstrated that through this prioritization, the stability and robustness of a quadrotor is increased in case of motor-input saturations.

References

[1] M. Hehn and R. D'Andrea, "A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers," *J. of Mechatronics*, vol. 24, no. 8, pp. 954 – 965, 2014.

[2] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. J. of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[3] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 1722–1729.

[4] 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," *J. of Mechatronics*, vol. 24, no. 1, pp. 41–54, Feb. 2014.

[5] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro UAV testbed," *IEEE Robotics Automation Magazine*, vol. 17, no. 3, pp. 56–65, Sept. 2010.

[6] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3D mapping with a quadrotor MAV," *J. of Field Robotics*, pp. 1556–4967, 2015.

[7] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, Sept. 2004, pp. 2451–2456.

[8] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on SO(3)," in *IEEE Conf. on Control Applications (CCA)*, Sept. 2015, pp. 1160–1166.

[9] Y. Yu, S. Yang, M. Wang, C. Li, and Z. Li, "High performance full attitude control of a quadrotor on SO(3)," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2015, pp. 1698–1703.

[10] Y. Kawai and K. Uchiyama, "Design of frequency shaped LQR considering dynamic characteristics of the actuator," in *IEEE Intl. Conf. on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 1235–1239.

[11] B. Wang, K. A. Ghamry, and Y. Zhang, "Trajectory tracking and attitude control of an unmanned quadrotor helicopter considering actuator dynamics," in *IEEE Chin. Control Conf. (CCC)*, July 2016, pp. 10 795–10 800.

[12] T. Lee, M. Leoky, and N. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *IEEE Conf. on Decision and Control (CDC)*, Dec. 2010, pp. 5420–5425.

[13] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2011, pp. 2520–2525.

[14] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. of the Int. Symp. of Robotics Research (ISRR)*, 2013.

[15] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[16] M. Hehn and R. D'Andrea, "Real-time trajectory generation for quadrocopters," *IEEE Trans. Robotics*, vol. 31, no. 4, pp. 877–892, 2015.

[17] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *IEEE Eur. Control Conf. (ECC)*, 2013, pp. 1383–1389.

[18] J. C. Monteiro, F. Lizarralde, and L. Hsu, "Optimal control allocation of quadrotor UAVs subject to actuator constraints," in *IEEE Am. Control Conf. (ACC)*, July 2016, pp. 500–505.