Simple Controller Tuning for Unmanned Aerial Vehicles using Governors

1st Raphael Dyrska
Automatic Control and Systems Theory
Ruhr-Universität Bochum
Bochum, Germany
raphael.dyrska@rub.de

1st Jens Müller
Automatic Control and Systems Theory
Ruhr-Universität Bochum
Bochum, Germany
jens.mueller-r55@rub.de

2nd Miroslav Fikar
Institute of Information Engineering, Automation, and Mathematics
Slovak University of Technology in Bratislava
Bratislava, Slovakia
miroslav.fikar@stuba.sk

3rd Martin Mönnigmann
Automatic Control and Systems Theory
Ruhr-Universität Bochum
Bochum, Germany
martin.moennigmann@rub.de

***Abstract*—A simple governor-based controller tuning is implemented and tested for the application to unmanned aerial vehicles (UAVs). We show that the governor-based tuning approach enables the high-level controller tuning of the elaborate controller structure of the UAVs by using a single tuning parameter only. We implement the approach on two UAV platforms of different sizes and controller frameworks to prove the versatility of the given approach. The validation is performed with experimental data collected for the UAVs flying in the supervised environment of an indoor flight laboratory.**

***Index Terms*—unmanned aerial vehicle, controller tuning, governor**

I. Introduction

Unmanned aerial vehicles (UAVs) are used in an increasing number of applications, ranging from crop monitoring and spraying in agriculture over automatic inspection of infrastructure to search and rescue tasks (see, e.g., [1]–[3]). The field of possible applications grows even further when combining UAVs with ground robotics (see, e.g., [4] for a literature review in the context of production systems and the references therein). The hardware of UAVs is commonly tailored to specific tasks due to the broad spread of possible applications. Thus, UAVs come in various sizes and are equipped with different onboard sensors and actuators meeting the specific needs of the desired application. Consequently, UAVs are made available from many different manufacturers and with a heterogeneous set of possible hardware configurations.

While UAVs were mainly operated by a minority of experts and enthusiasts a few years ago, the increased ease of operation and the increased robustness of the integrated controller structure enable almost everyone to control UAVs after a short briefing. This stems from the fact that the controller hardware is more powerful nowadays and undertakes tasks that previously had to be manually performed by an expert operator. Consequently, the direct steering of UAVs is more and more replaced by inputting reference values only for the closed-loop controller running on the UAV itself. Thus, the flight behavior of UAVs is mainly determined by the tuning of the control loops running on the flight controller.

The increased ease of operation masks the complexity of the controller structure usually running unnoticed by the operator. As the overall controller structure contains multiple control loops with a large number of parameters, the accurate tuning of the controller towards certain control goals necessarily requires expert knowledge and bears the risk of accidents.

An incorrect tuning may result in significant damage up to a total loss of the UAV or even cause damage to third parties. In some cases, a direct tuning of the control parameters may also require access to the software scripts implemented on the flight controller. Since these may be of enormous financial interest to the manufacturer, such an access is often not granted. Additionally, a retuning of the parameters or other significant changes to the control framework may cause conflicts regarding the UAV insurance. As a consequence, many UAV platforms do not provide an easy way for tuning the controller structure nor do they grant access to the controller implementation.

Although the majority of commercially available UAVs are already delivered with controller parameters that yield stable flight behavior, the operator might want to change the controller tuning according to specific needs. As an example, a UAV operated outdoors may be tuned more aggressively to travel large distances in less time compared to a UAV operated in constricted indoor environments. A UAV operated indoors will usually be tuned more conservatively, resulting in slower movements and higher control accuracy to prevent contact with surfaces and equipment. Figure 1 shows such a scenario in

This paper is funded by the European Union's Horizon Europe under grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries) and by the Alexander von Humboldt Foundation research group linkage cooperation program.

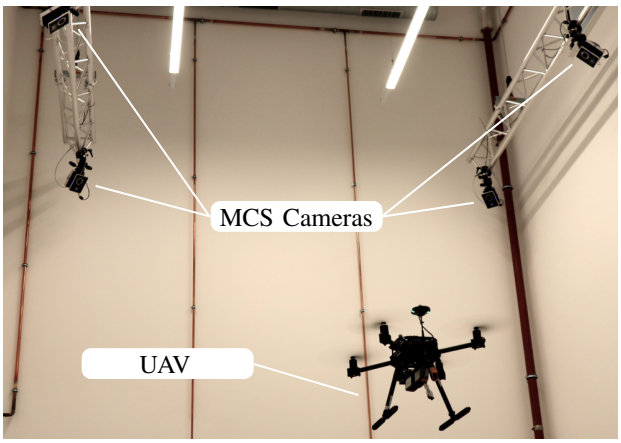

Fig. 1. UAV flying in the flight laboratory inside the research building ZESS of the Ruhr-University Bochum, Germany, equipped with Cameras of the Motion Capture System.

the flight laboratory inside the research building ZESS of the Ruhr-University Bochum operated by the authors.

The complexity of the (potentially not even accessible) UAV controller structure on the one hand and the need to retune the controller with respect to specific use cases of the UAV (without expert knowledge) on the other hand are addressed in this paper. We use a previously presented governor-based tuning approach to retune UAVs without changing any internal tuning parameters. This way the controller structure and its parameters of the UAV remain unchanged, and only the reference values transmitted to the UAV are modified. As a result, the approach is highly adaptable to UAV platforms of different sizes regardless of the specific controller structure implemented by the manufacturer.

Section II introduces the governor-based approach and the basic terminology. Section III presents the experimental setup and the specific UAV platforms used in this contribution. Section IV gives results of flight experiments and evaluates the performance of the given approach regarding the implementation on the UAVs. Conclusions are given in Section V.

II. Tuning by Governors

We briefly recall the main idea of using governors as summarized, e.g., in [5] (or, equivalently, [6]), and present the different variants suitable for tuning the control behavior.

To tune the controller used in a UAV without changing internal parameters, we focus on the control error

$$e(t) = w(t) - y(t), \tag{1}$$

with $w(t)$ and $y(t)$ being the reference and the output value, respectively. The control error $e(t)$ serves as the input of the original controller. A simple way to modify the control error artificially is to introduce a static scalar K, such that the control error $e(t)$ can be manipulated according to

$$\bar{e}(t) = Ke(t) = K(w(t) - y(t)). \tag{2}$$

A new control error $\bar{e}(t)$ results that enforces a different control signal when forwarded to the implemented controller. This so-called Error Governor can be used to generate more aggressive or more conservative control signals by choosing the value of K larger or smaller than one. $K = 1$ yields original control behavior.

Error Governors are useful especially if the control error is computed externally and forwarded to the controller. If the control error is computed directly on the controller hardware, the modification of $e(t)$ might be difficult. Alternatively, the reference value and the process output can be modified before being sent to the controller according to

$$\bar{w}(t) = Kw(t), \quad \bar{y}(t) = Ky(t),$$

which results from (1) by introducing the manipulated reference value and process output $\bar{w}(t)$ and $\bar{y}(t)$, respectively, describing the manipulated error by

$$\bar{e}(t) = \bar{w}(t) - \bar{y}(t). \tag{3}$$

However, it might be more practicable to manipulate only one of the signals forwarded to the controller. The same modified control error $\bar{e}(t)$ as in (1) can be achieved by changing, e.g., the reference value $w(t)$ using K and leaving the output signal $y(t)$ unchanged. This so-called Reference Governor then only modifies $w(t)$ and results from inserting $\bar{e}(t) = K(w(t) - y(t))$ in (3) and using $\bar{y}(t) = y(t)$. The new reference value $\bar{w}(t)$ reads

$$\bar{w}(t) = Kw(t) + (1 - K)y(t). \tag{4}$$

If sent repeatedly to the UAV, the Reference Governor thus updates the reference value with respect to the current output value $y(t)$ until $\bar{w}(t) = w(t)$ holds for $y(t) = w(t)$.

Analogously, a so-called Output Governor can be implemented by modifying the current output value depending on the current reference value

$$\bar{y}(t) = Ky(t) + (1 - K)w(t). \tag{5}$$

For the experiments presented in the following section, we chose the Reference Governor structure given in (4), since the control error is computed onboard by the UAVs controlled here. Thus, the control error is not directly accessible outside the UAV.

III. Experimental Setup

We apply the Reference Governor to two UAVs of fundamentally different sizes, i.e., a Crazyflie 2.1 and a customized variant of the Holybro X500. Figure 2 illustrates the sizes of the UAVs, which are introduced in detail in the following subsections. The control strategies of both UAVs are implemented within different frameworks, rendering a unified tuning strategy on an implementational level difficult, if not impossible.

We use a Motion Capture System (MCS) for the location of both, the Crazyflie and the X500. The MCS consists of 10 Vicon Vantage V5 infrared cameras and uses the software

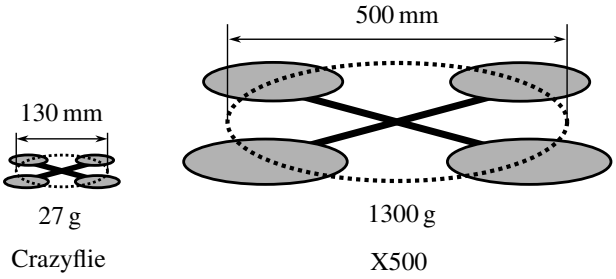

Fig. 2. Sketch of the used UAVs illustrating their different sizes and weights.

Tracker 3 for computing the positions of the UAVs with a precision of up to 1 mm. The UAVs are equipped with a unique and asymmetric arrangement of reflective markers defining the rigid body objects to be captured. Figure 1 shows a UAV used for validation purposes flying in the flight laboratory with the infrared cameras visible in the background.

A. *Crazyflie*

The first UAV to be tuned is the Crazyflie 2.1 from bitcraze. The open-source small-scale quadrotor with a takeoff weight of around 27 g and a rotor-to-rotor diameter of approximately 130 mm is based on an STM32F405, and there is a wide range of existing implementations and various extension possibilities. For an introduction to the project, we refer to [7], presenting the former version Crazyflie 2.0. The Crazyflie is a popular choice to test new control strategies due to the small dimensions and the resulting small damage potential. Recent research deals, e.g., with the implementation of various control methods such as Fuzzy control [8], Nonlinear Model Predictive Control [9], and Reinforcement Learning [10], or with the combination of different control frameworks [11]. Figure 3 shows the Crazyflie with reflective markers for the MCS mounted on a 3D-printed expansion deck.

For the control of the Crazyflie in combination with the MCS as positioning feedback, we modify parts of the CrazyFlie-SdGN[1] Github Project. Here, high-level control commands are used, e.g., to realize a point-by-point flight based on pre-specified coordinates.

The control structure of the Crazyflie is sketched in Fig. 4. The scheme is a modification of the original figure presented in the bitcraze documentation[2], extended by the MCS as used by CrazyFlie-SdGN and the Reference Governor proposed in Section II (red components). All control loops of the cascaded PID scheme are shown with the sampling rates and corresponding inputs and outputs. The MCS serves as additional outer position feedback used by the Extended Kalman Filter (EKF) to estimate all necessary feedback signals.

Ignoring the red parts, Fig. 4 shows that the Crazyflie is controlled according to the tuning of the PID parameters of

[1]https://github.com/slim71/CrazyFlie-SdGN

[2]https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/controllers/

Fig. 3. Crazyflie 2.1 quadrotor with reflective markers mounted on a 3D-printed expansion deck.

each control loop. Thus the tuning of the overall flight behavior requires the tuning of multiple individual control loops.

By manipulating only the desired position according to the Reference Governor (4) that is forwarded to the high-level controller and that enters the cascaded PID scheme from the far left (red part in Fig. 4), we are able to tune the flight behavior without modifying any of the inner controllers. As expressed by (4), both, the original reference value as well as the current position are needed for manipulating the reference value. Here, we use the current position of the Crazyflie measured by the MCS.

We chose a sampling rate of 10 Hz for sending manipulated reference values, showing satisfying results as presented in Section IV. The communication between the Crazyflie and the guiding PC is established via the Crazyradio PA USB radio dongle using a 2.4 GHz ISM band radio.

B. *Holybro X500*

We use the customized quadrotor based on the Holybro X500 shown in Fig. 5 as the second UAV to be tuned. This UAV has a rotor-to-rotor diameter of 500 mm and is able to carry an additional load of up to 1 kg, rendering it suitable for carrying material, various additional sensors, and cameras. Consequently, the X500 covers fundamentally different application scenarios than the Crazyflie. The X500 is controlled by a Pixhawk Cube Orange flight controller, running the open-source autopilot software system Ardupilot[3], which is commonly used for many UAV applications. In our setup, we established the communication to the X500 using the Hex Herelink 1.1 controller, again operating on a 2.4 GHz ISM band.

The X500 is meant to be operated outdoors and uses a top-mounted GPS antenna visible in Fig. 5 as an external position feedback. We replace the GPS position information with the position information provided by the MCS. This way, the controller structure of the X500 is not changed and the

[3]https://ardupilot.org/

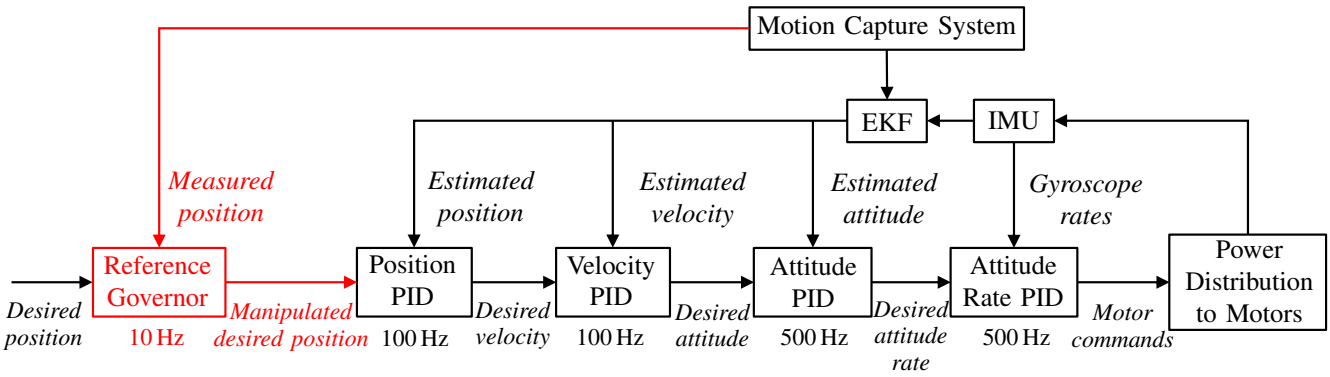

Fig. 4. Cascaded PID control scheme of the Crazyflie 2.1, extended by the Reference Governor as in (4) highlighted in red.

Fig. 5. Customized Holybro X500 with a Pixhawk Cube Orange flight controller, external receiver for position data, and reflective markers.

proposed approach would yield comparable results when using GPS information.

We use the MAVProxy[4] software with the associated Vicon module to convert the position information coming from the MCS to GPS information and periodically send it to the X500. The GPS information is sent via an additional WiFi connection deployed using an ESP8266 Microcontroller and the MAVESP8266 firmware[5]. The described setup follows the instructions given in the Ardupilot documentation.[6]

Similar to the Crazyflie setup, we manipulate the desired position by using (4), again sending the manipulated reference value with a sampling rate of 10 Hz. The Reference Governor is implemented in a MAVProxy module and resembles the Crazyflie implementation. This underlines the simple applicability of the Reference Governor to various controller frameworks without additional implementation effort.

[4]https://ardupilot.org/mavproxy
[5]https://github.com/dogmaphobic/mavesp8266
[6]https://ardupilot.org/copter/docs/common-vicon-for-nongps-navigation.html

C. Experiments

We generate step responses for a change in the reference height for the Crazyflie and the X500 to analyze the suitability of a Reference Governor as a tuning mechanism for different UAVs with different controller hardware. We set the default height to 1 m instead of starting from the ground to exclude disturbances due to the ground effect. The step is a change of the reference value $w(t)$ to a new reference value of 2 m in direction of the z-axis, resulting in a step length of $\Delta w = 1\,\mathrm{m}$.

Starting with a tuning factor $K = 1$ (see (4) in Sect. II), we first measure a benchmark step response. Subsequently, we change the factor K in both directions, starting with an increase to generate a more aggressive control behavior ($K > 1$). Afterwards, we chose $K < 1$ to yield a more conservative flight behavior.

The Crazyflie runs on a single-cell lithium polymer battery and the flight behavior is known to be affected by the battery voltage [12]. We replace the battery with a fully charged one after each Crazyflie flight and before changing the factor K to neglect the influences of different battery charge levels in our experiments.

The X500 is powered by a four-cell lithium polymer battery. The impact of the voltage on the flight behavior is smaller for the X500. Consequently, all tests are conducted with one battery charge.

We analyze all step responses for both UAVs quantitatively using well-established parameters:

- Rise time t_r: time span from changing the reference value to entering a tube of 5% tolerance around the reference value for the first time
- Settling time t_s: time span from changing the reference value to entering a tube of 5% tolerance around the reference value for the last time
- Maximum overshoot e_max: difference between maximum output y_max and new reference value w with respect to the reference change Δw as in

$$e_\mathrm{max} = \frac{y_\mathrm{max} - w}{\Delta w} \tag{6}$$

The $5\,\%$ tolerance band around the reference value is introduced to neglect the influence of small disturbances that are

inevitable during the experiments. The results for both UAVs and their evaluation are presented in Section IV.

IV. Experimental results

We first analyze the results of the experiments for both UAVs in a qualitative fashion and corroborate the observations using the quantitative parameters introduced in Section III-C afterward.

A. Results for the Crazyflie

Figure 6 shows the step responses of the Crazyflie to a change in the reference height and three different tuning factors K. According to (4), the original reference value results for $K = 1$ (solid black line). The Crazyflie slightly overshoots the new reference height of 2 m (dashed black line) before minimizing the remaining control error. As expected, a faster rise to the reference, by tolerating a larger overshoot, is observable for a more aggressive controller tuning using a tuning factor of $K = 1.5$ (yellow line). The reference value is artificially increased according to (4) until the actual reference of 2 m is reached. The tuning towards a less aggressive behavior ($K = 0.66$, purple line) results in a more damped step response as expected. The resulting controller is more conservative and yields a reduced overshoot, however, by accepting to reach the reference value significantly later.

A more quantified comparison is conducted using the results summarized in Table I. All parameters, the rise time t_r, settling time t_s, and maximum overshoot e_max underline the qualitative observations presented before. For a tuning factor of $K = 1.5$, the rise time decreases by around $6.2\,\%$ with respect to the original tuning ($K = 1$), while both, the settling time and maximum overshoot increase by around $5.9\,\%$ and $53\,\%$, respectively. In contrast, for a more conservative controller tuning using the Reference Governor, an increase of the rise time by almost $110\,\%$ can be observed, while the maximum overshoot decreases by around $27\,\%$. However, the conservative controller still overshoots the reference value. Due to the overshoot in combination with the conservative flight behavior, the settling time of the conservative controller increases by around $8.8\,\%$. This is, however, not very intuitive and underlines the fact that the governor enables a simple and relative tuning, that still depends on the internal controller design.

TABLE I
Results for the Crazyflie.

K	0.4	1	1.5
rise time t_r	2.3877 s	1.1398 s	1.0687 s
settling time t_s	7.8447 s	7.2090 s	7.6350 s
max. overshoot e_max	0.1102	0.1511	0.2312

B. Results for the X500

We use the same step function and the same tuning factors for the X500 experiments that we used for the experiments with the Crazyflie. This way, a qualitative comparison between both UAVs is possible in addition to the evaluation of each experiment. Figure 7 shows the step responses of the X500 using $K = \{0.4, 1, 1.5\}$.

It is visible from Fig. 7 that the qualitative tuning of the controller using the tuning factor K is successful. The reference tuning ($K = 1$) is depicted as a solid black line in Fig. 7. Similar to the tuning of the Crazyflie, a more aggressive tuning can be achieved with $K = 1.5$ (yellow line), while $K = 0.4$ yields a more conservative flight behavior (purple line).

The quantitative results of the X500 experiments are summarized in Table II. It follows from the results in Table II

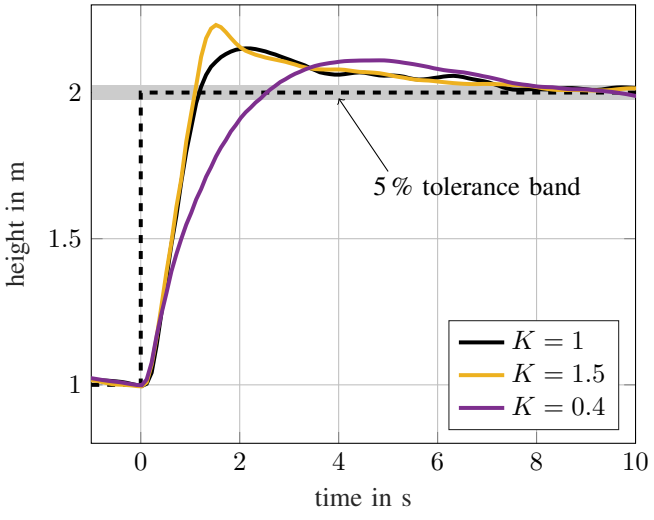

Fig. 6. Step responses of the Crazyflie for the regular controller (black) and a more (yellow) respectively less (purple) aggressive controller tuning. The reference value w is shown by the dashed line.

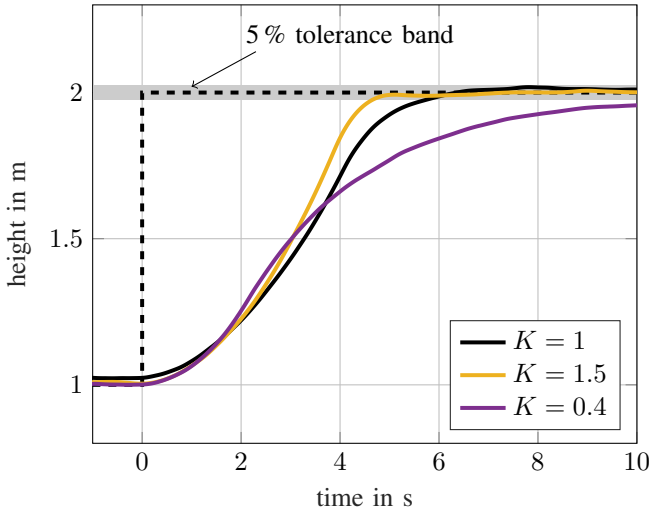

Fig. 7. Step responses of the X500 for the regular controller (black) and a more (yellow) respectively less (purple) aggressive controller tuning. The reference value w is shown by the dashed line.

TABLE II
RESULTS FOR THE X500.

K	0.4	1	1.5
rise time t_r	n.a.	5.7347 s	4.6532 s
settling time t_s	n.a.	5.7347 s	4.6532 s
max. overshoot e_max	n.a.	n.a.	n.a.

that the rise time can be reduced by around $18\,\%$ in comparison to the reference tuning by tuning the controller with $K = 1.5$. However, a more detailed quantitative evaluation is not possible for the X500 experiments, because not all values can be obtained from Fig. 7. For instance, all tuning factors yield aperiodic controllers without overshooting the reference value. Additionally, the controller retuned with $K = 0.4$ does not reach the reference value within the experiment time of 10 s.

Comparing Fig. 6 and Fig. 7 reveals that using the same tuning factors $K = \{0.4, 1, 1.5\}$ yields fundamentally different control behavior for the tested UAVs. While all measurements with the Crazyflie overshoot the reference value, the experiments performed with the X500 present an aperiodic step response. This underlines the fact that the retuning always occurs relatively to the present controller of the specific UAV. Although this might seem obvious, it must be taken into account when starting the retuning procedure of a UAV.

However, tuning the controller even more aggressively and provoking the overshooting of the reference value is also possible for the X500. Figure 8 shows the step responses of the X500 with $K = \{1, 2\}$. For $K = 2$, the tuning is aggressive enough such that the X500 also overshoots the reference value. Consequently, similar results can be achieved for the Crazyflie and the X500. Obviously, specific step responses correspond to different tuning factors, and the correct tuning factor yielding the desired flight behavior must be found in an iterative fashion.

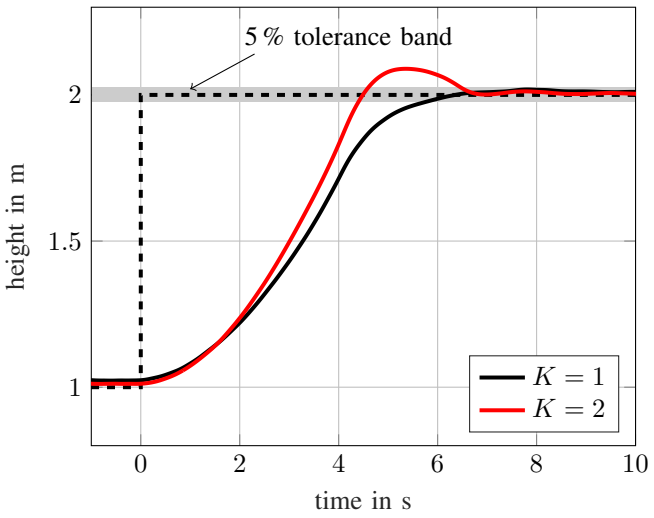

Fig. 8. Step responses of the X500 for the regular controller (black) and a more aggressive (red) controller tuning. The reference value w is shown by the dashed line.

V. CONCLUSION

We showed that despite their complex controller frameworks, the control behavior of different UAVs can suitably be retuned with a single tuning parameter using governors. We implemented a Reference Governor to tune the controllers of two UAVs of different scales and showed with several experiments that the governor approach yields the desired effect for both systems. Thus, the presented approach is suitable to retune UAVs whenever the detailed retuning of internal control parameters is not possible due to lacking access or prohibitive restrictions.

While this work investigated the tuning for reference changes of the height, future work will focus on validating the retuning for all degrees of freedom of the UAV.

REFERENCES

[1] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of uav applications for precision agriculture," *Computer Networks*, vol. 172, p. 107148, 2020.

[2] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse, "State-of-the-art technologies for uav inspections," *IET Radar, Sonar & Navigation*, vol. 12, no. 2, pp. 151–164, 2018.

[3] M. Silvagni, A. Tonoli, E. Zenerino, and M. Chiaberge, "Multipurpose uav for search and rescue operations in mountain avalanche events," *Geomatics, Natural Hazards and Risk*, vol. 8, no. 1, pp. 18–33, 2017.

[4] J. Sinnemann, M. Boshoff, R. Dyrska, S. Leonow, M. Mönnigmann, and B. Kuhlenkötter, "Systematic literature review of applications and usage potentials for the combination of unmanned aerial vehicles and mobile robot manipulators in production systems," *Production Engineering*, pp. 579–596, 2022.

[5] M. Fikar, K. Kiš, K. M., and M. Mnnigmann, "Simple tuning of arbitrary controllers using governors," in *submitted to the IFAC World Congress 2023*, 2023.

[6] M. Fikar, M. Klaučo, and K. Kiš, "A general controller tuning using governors," 2022, Technical Report (online). https://www.uiam.sk/assets/publication_info.php?id_pub=2366.

[7] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, 2017, pp. 37–42.

[8] F. Candan, A. Beke, and T. Kumbasar, "Design and deployment of fuzzy pid controllers to the nano quadcopter crazyflie 2.0," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018, pp. 1–6.

[9] B. B. Carlos, T. Sartor, A. Zanelli, G. Frison, W. Burgard, M. Diehl, and G. Oriolo, "An efficient real-time nmpc for quadrotor position control under communication time-delay," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2020, pp. 982–989.

[10] J. E. Kooi and R. Babuška, "Inclined quadrotor landing using deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2361–2368.

[11] M. Nithya and M. Rashmi, "Gazebo-ros-simulink framework for hover control and trajectory tracking of crazyflie 2.0," in *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, 2019, pp. 649–653.

[12] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. J. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.