MPC Flight Control for a Tilt-Rotor VTOL Aircraft

L. Bauersfeld, L. Spannagl, Guillaume Ducard, Christopher Onder

This is the accepted version of the paper : L. Bauersfeld, L. Spannagl, G. Ducard, C. Onder, "MPC Flight Control for a Tilt-rotor VTOL Aircraft," in IEEE Transactions on Aerospace and Electronics Systems, Vol. 57, Issue 4, August 2021, pp. 2395 – 2409. 15 pages.
https://doi.org/10.1109/TAES.2021.3061819

MPC Flight Control for a Tilt-Rotor VTOL Aircraft

LEONARD BAUERSFELD
LUKAS SPANNAGL
Institute for Dynamic Systems, and Control, ETH Zürich, Switzerland

GUILLAUME J. J. DUCARD
University Cote d'Azur, Centre national de la recherche scientifique, I3S, Sophia Antipolis, France

CHRISTOPHER H. ONDER
Institute for Dynamic Systems, and Control, ETH Zürich, Switzerland

This article presents a model predictive control (MPC) controller and its novel application to a hybrid tilt-quadrotor fixed-wing aircraft, which combines vertical takeoff and landing (VTOL) capabilities with high-speed forward flight. The developed MPC controller takes a velocity command from the pilot and then computes optimal attitude setpoints and propeller-tilt angles that are supplied to a fast inner attitude controller. A control allocation algorithm then maps the output of the inner attitude loop to actuator commands. The proposed MPC and control allocation of this article constitute a unified nonlinear control approach for tilt-rotor VTOL aircraft, valid in all flight modes and transitions in between. The whole approach is verified both in simulations and in real-world outdoor experiments with a remote controlled VTOL aircraft transitioning from hover to high speed and vice versa in a stable and controlled manner. Results show superior performance compared to the common binary-switch transition strategy between multicopter flight mode and the fixed-wing flight mode. The MPC controller also consistently performs better than a previously developed fused-PID control architecture in our tests.

Authors' addresses: Leonard Bauersfeld, Lukas Spannagl, and Christopher H. Onder are with the Institute for Dynamic Systems, and Control, ETH Zürich, 8092 Zürich, Switzerland, E-mail: (leonardb@student.ethz.ch; spalukas@student.ethz.ch; onder@idsc.mavt.ethz.ch); Guillaume J. J. Ducard is with the I3S, University Cote d'Azur, Centre national de la recherche scientifique, 06903 Sophia Antipolis, France, E-mail: (ducard@i3s.unice.fr). *(Corresponding author: Leonard Bauersfeld.)*

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have become increasingly popular among researchers, industry professionals and hobbyists for carrying out a variety of tasks ranging from inspection of power lines to racing competitions. Most vehicles can either be categorized as fixed-wing aircraft or multirotor aircraft (multicopters). Fixed-wing aircraft generate the required lift with their wings whereas multirotor aircraft have upwards pointing propellers generating the force needed to support their weight. Fixed-wing aircraft are very efficient during high-speed long-distance flight. However, despite being much less energy efficient, multicopters can operate in tight spaces. A vertical takeoff and landing (VTOL) aircraft with wings and tilting propellers combines the advantages of both aircraft designs. The key design aspect is that the propellers can be tilted upward for takeoff and landing (multicopter configuration) and can be tilted forward for long-distance flight (fixed-wing configuration). Fig. 1 shows such a tilt-rotor VTOL aircraft.

Controlling hybrid vehicles like the tilt-rotor VTOL aircraft is challenging because the vehicle's dynamics in a fixed-wing configuration are very different to those in a multicopter configuration. To illustrate this, consider the situation, where the vehicle is flying at low forward speed and tasked to accelerate and climb: a) a multicopter needs to pitch down to accelerate and increases the thrust of its propellers to climb, whereas b) a fixed-wing aircraft needs to pitch up and to increase thrust to execute the same maneuver. Another challenge stems from the fact that a tilt-rotor VTOL aircraft is overactuated, which means it has more actuators (rotors, control surfaces) than degrees of freedom. The process of mapping the desired forces and torques to actuator outputs is thereafter referred to as *control allocation*. To fully use the potential of the tilt-rotor VTOL aircraft, not only a smooth transition from the multicopter configuration to the fixed-wing configuration is important, but the use of all in-between flight states (partially tilted propellers) must be possible as well.

The main contribution of this article is the development and the successful flight testing of a model predictive control (MPC) flight control system for a tilt-rotor VTOL aircraft, such as the one shown in Fig. 1. Although MPC is a known control approach, its integration and formulation to the problem of tilt-rotor VTOL aircraft, as a unified control approach, i.e., valid in all flight modes, is new. Indeed, to the best of our knowledge, this article is the first to report successful real-world outdoor flight in extensive experiments for a tilt-rotor VTOL UAV with the following.

1) A smooth and seamless transition from hover to high-speed flight and vice versa, autonomously performed by the flight controller.
2) A flight controller that is able to control the tilt-rotor VTOL aircraft throughout its whole flight envelope, without the need to switch between several flight controllers.
3) A flight stack (control allocator, low-level control, MPC) that runs in real-time and onboard the vehicle.

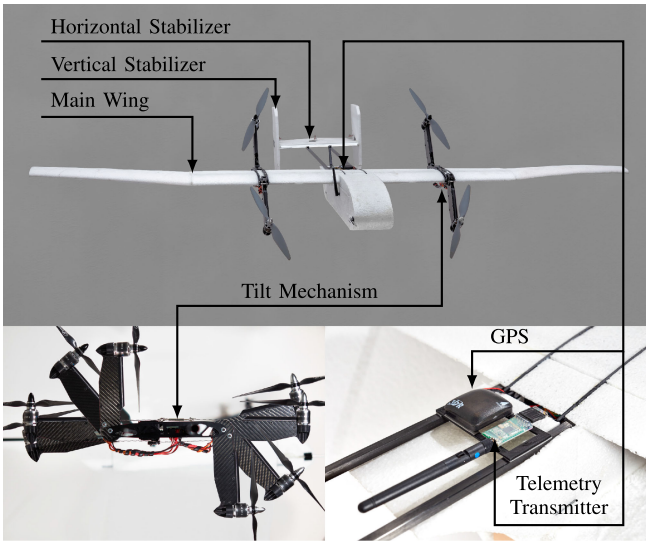

Fig. 1. Tilt-rotor VTOL aircraft used in this work is from *WingCopter*. The design features four tilting propellers where the propellers on the same side of the fuselage are always tilted together. Additionally, a pitot tube, a GPS receiver, a telemetry transmitter and a Pixhawk autopilot were fitted to the vehicle.

These achievements were possible due to the specific MPC formulation used as follows:

1) the design of a cost function with soft constraints custom-tailored to the vehicle at hand;
2) the use of a soft ℓ_1-norm tracking cost to achieve fast computation times;
3) the choice of a suitable receding horizon compatible with the vehicle dynamics and available computational power onboard the vehicle's small form-factor computer;
4) an efficient-to-evaluate nonlinear model of the vehicle dynamics.

The presented control approach also uses a custom control allocator (CA) originally developed in [1]. Overall, satisfactory and reliable flight performance have been successfully demonstrated in outdoor real flights at speeds up to 22 m/s.

The remainder of this article is organized as follows. First, a review of common vehicle designs and control strategies for VTOL aircraft is presented. Then, an overview of the control architecture is given followed by a detailed discussion of its components in the subsequent sections. Emphasis is placed on the choice of equations, states, and constraints. Finally, Section VI presents the experiments in simulation and with the real vehicle.

II. RELATED WORK

Research about VTOL UAVs is a broad topic due to the various existing vehicle designs. The vehicles can roughly be divided into three categories. 1) Tilt-rotor, 2) tilt-wing, and 3) tail-sitter vehicles. In the case of the first two, the fuselage remains horizontal during the vertical takeoff and landing as well as during forward flight. Tail-sitters on the other hand transition back and forth between a vertical (takeoff, landing and hover) and a horizontal (forward flight) configuration.

Some recent control approaches [2], [3] split the control problem into several control regimes with discrete switches between them and focus on the transition phase between the individual modes. For instance, Li *et al.* [3] controlled the transition phase of a tail-sitter using optimization and Liu *et al.* [2] applied robust nonlinear control. MPC has also been successfully applied to a ducted tail-sitter in [4]. The tail-sitter described in [5] uses a lookup table of optimal control inputs to implement an MPC-like control architecture. Verling *et al.* [6] used a combination of model-based optimization and an attitude controller to solve the same problem. Overall, the tail-sitter type vehicle has been well studied in the MPC context, both theoretically and in practice.

The tilt-wing vehicles have their propellers rigidly attached to their tiltable wings. Tran *et al.* [7] utilized a robust control augmentation system to improve the transition behavior of a quadrotor-tilt-wing VTOL aircraft and validate their design on a real vehicle. MPC has also been applied to such a system in [8]. This approach uses a linearized model and only simulation experiments were presented. The third type of VTOL UAVs—tilt-rotors—represent a very popular field of research. Various geometries with two to six propellers and with and without wings have been studied. $\mathcal{H}_\infty$ and MPC have been applied to bicopters in [9] and [10], respectively. Ryll *et al.* [11] designed a control framework for a fully actuated quad-tilt-copter. In [12] and [13], two more types of fully actuated multicopters are developed. Because they are fully actuated, position and orientation can be decoupled. Ryll *et al.* [14] described a control approach that is capable of switching between an underactuated and a fully actuated configuration with just a single additional actuator which tilts the propellers.

The vehicle described in this article is also a tilt-rotor vehicle, but has the aerodynamic actuators in addition to the tiltable propellers. This category is called tilt-rotor aircraft. A common approach for VTOL UAVs in general and especially tilt-rotor aircraft is to use two or three separate controllers [15], [2], [16], [10], [3], a multicopter controller, a fixed-wing aircraft controller and a transition controller. This approach is very prominent for tilt-rotor aircraft because the multicopter and fixed-wing controllers are well researched. One then needs to switch between the two flight configurations, and the most simple solution is a preprogrammed feed-forward transition controller, as done on the commercial PixHawk autopilot. Such open-loop controllers are not robust and research on how to improve them dates back to 1971 when Nardizzi *et. al* [17] studied how to perform transitions in minimum time with open and closed-loop control.

Mehra *et al.* [18] applied MPC to an XV-15 aircraft, but has to use fuzzy logic to make the problem computationally feasible and does not show experimental results. Tilt-duct

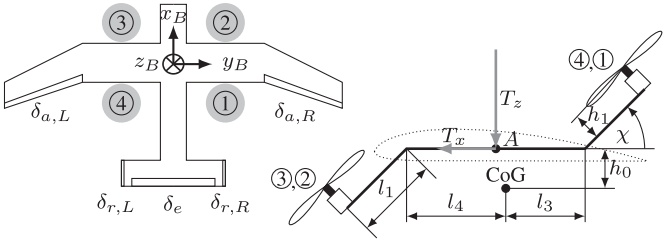

Fig. 2. Front-right-down body frame is attached to the vehicle. The aircraft has five aerodynamic actuators, namely two ailerons, two rudders, and one elevator. The propeller-tilt angle is measured w.r.t. the body x-axis and the propellers on the same side of the fuselage tilt simultaneously.

type aircraft are analyzed in [19] and [20], where [20] also shows experimental results, but does not apply MPC. Allenspach *et al.* [21] applied MPC to a quadrotor-tilt aircraft but do not provide experimental results. Finally, Papachristos *et al.* [22] applied MPC to improve position control near hover and provide experimental results. Their setting, however, is quite different compared to ours as they use MPC to improve the disturbance rejection near hover and do not focus on fast forward flight or a transition.

Overall, the body of research is focused on the mathematical modeling of VTOL aircraft and the subsequent controller synthesis based on those models. For tail sitters, MPC has been successfully developed and validated on a series of real vehicles. For tilt rotor vehicles, however, the research so far only either develops an MPC and does not validate it in experiments on a real vehicle or validates control strategies that are not based on optimization. To the best of the authors' knowledge, the only experimentally validated MPC on a tilt rotor UAV was specifically designed to reject position disturbances near hover [22]. Therefore, a next step to further advance tilt-rotor VTOL research is to apply MPC to the considered vehicle and validate the approach for all flight scenarios. This is the goal of this article: to present a unified control approach that can be used throughout the whole flight envelope of the tilt-rotor VTOL aircraft.

III. OVERVIEW

A. Notation and Coordinate Systems

In this article, vector quantities are represented by bold symbols. Most calculations are carried out in the front-right-down frame that is attached to the vehicle, as shown in Fig. 2, and it is denoted by a subscript "B" in ambiguous cases. The deflection angles of the aerodynamic actuators are given by δ_a, δ_e, and δ_r for the aileron, the elevator, and the rudder, respectively. The left and right propeller pair (e.g., ①, ② and ③④) can be tilted independently. The tilt angles χ_L and χ_R of the left and right propeller pair are measured w.r.t. the body x-axis x_B. This means that $\chi_L = \chi_R = 0$ corresponds to "multicopter mode" whereas $\chi_L = \chi_R = \frac{\pi}{2}$ corresponds to the fixed-wing aircraft configuration. The total thrust of all four propellers is denoted by $\mathbf{T}$. Due to the geometry of the vehicle, a thrust in the body-y_B direction cannot be generated

$$\mathbf{T} = \begin{bmatrix} T_{x_B} & 0 & T_{z_B} \end{bmatrix}^\top. \tag{1}$$

To control the attitude of the vehicle, torques around the x_B, y_B, and z_B axes are required. Such torques can be generated by the propellers as well as by deflecting the aerodynamic actuators, The vector of all torques is defined as

$$\boldsymbol{\tau} = \begin{bmatrix} L & M & N \end{bmatrix}^\top$$

where L is the roll torque, M is the pitch torque, and N is the yaw torque.

The inertial or global frame used in this work is a north-east-down (NED) frame. The attitude $\boldsymbol{\Psi}$ of the vehicle relative to that inertial frame is described by the three Euler angles roll ϕ, pitch θ, and yaw ψ

$$\boldsymbol{\Psi} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top.$$

The velocity $\mathbf{v}$ is given in the inertial frame whereas $\mathbf{v}_B$ is given in the body frame. Let $\mathbf{R}_\xi(\alpha)$ denote the rotation matrix around the axis ξ by an angle α. Then, the following relation holds:

$$\mathbf{v} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\mathbf{v}_B = {}_G\mathbf{R}_B\mathbf{v}_B, \quad \mathbf{v}_B = {}_G\mathbf{R}_B^\top\mathbf{v}. \tag{2}$$

B. Vehicle Specifications

The vehicle at hand is a modified *WingCopter v1* as shown in Fig. 1 that has been equipped with a GPS antenna, a three-axis accelerometer, three-axis gyroscope, and three-axis magnetometer, a telemetry transmitter, a pitot tube, a Pixhawk autopilot, and an Intel UpBoard small form-factor computer. The physical and aerodynamic specifications are listed in Table I. To obtain the moment of inertia of the vehicle around its principal axes, it was suspended by a metal wire. With a known torsional spring constant of the wire, the moment of inertia can be calculated from the oscillation frequency. The aerodynamic parameters were obtained with xFoil as detailed in [23].

The Pixhawk with the PX4 firmware is a very common autopilot among researchers and hobbyists alike as it open-source and can easily be adapted to the task at hand. The so-called "fused-PID" controller, the attitude controller and control allocation described in the following sections are custom-built modules that have been implemented into the firmware (based on version v1.10.0b4). The computer aboard the vehicle (referred to as companion computer) is an Intel UpBoard, which runs the MPC controller. It features an Intel Atom x5-Z8350 (4×1.44 GHz) processor and is running Ubuntu 18.04 LTS. It communicates with the Pixhawk autopilot using a serial interface. Popular alternatives to the Intel UpBoard would be the NVidia Jetson or Raspberry Pi 4.

TABLE I
Physical Parameters of the Aircraft

Name	Parameter	Value	Unit
Vehicle mass	m	2.7	kg
Inertia	I_{xx}	0.089	$\mathrm{kg\,m^2}$
	I_{yy}	0.067	$\mathrm{kg\,m^2}$
	I_{zz}	0.125	$\mathrm{kg\,m^2}$
Maximum total thrust	$T_{\max}$	48	N
Wingspan	b	2	m
Overall length	–	1.2	m
Maximum tilt-angle	$\chi_{\max}$	90	deg
Minimum tilt-angle	$\chi_{\min}$	-7	deg
Air density	ρ	1.2041	$\mathrm{kg/m^3}$
Surface area of main wing	S	0.4266	$\mathrm{m^2}$
Surface area of horizontal stabilizer	–	0.0465	$\mathrm{m^2}$
Surface area of vertical stabilizer	–	0.0744	$\mathrm{m^2}$
Effective area of fuselage	–	0.055	$\mathrm{m^2}$
Propeller yaw coefficient	C_Q	1.99017×10^{-7}	
Propeller thrust coefficient	C_T	1.11919×10^{-5}	
Aileron coefficient	C_{La}	0.11730	
Elevator coefficient	C_{Me}	0.55604	
Rudder coefficient	C_{Nr}	0.08810	
Tilting plane y-offset	L_0	0.29	m
Lever length	l_1	0.1575	m
Position of rear lever pivot	l_3	0.105	m
Position of front lever pivot	l_4	0.11	m
Tilting mechanism z-offset	h_0	0.015	m
Propeller height	h_1	0.05	m
Chord length	$\bar{c}$	0.2	m
Ramp function 1 slope parameter	a_{r1}	0.0185	
Ramp function 1 position parameter	b_{r1}	35.217	
Ramp function 2 slope parameter	a_{r2}	0.25	
Ramp function 2 position parameter	b_{r2}	2	

C. Control Architecture

After presenting the notation and the coordinate systems as well as the vehicle, this section gives an overview of the control architecture. The MPC controller and the control allocation will be explained in greater detail in the subsequent sections. The control architecture as a whole is summarized in Fig. 3.

1) *Velocity Loop:* The outer loop controller takes the vehicle's state and a velocity setpoint $\mathbf{v}_{\text{sp}}$ from the pilot through a RC transmitter as an input. A velocity setpoint is chosen for the outer loop for two reasons. 1) There is no straightforward way to map the RC-joystick space to the real vehicle's three-dimensional space. 2) The approach is general in the sense that any global path planner could be used to supply velocity setpoints and extend the controller. It outputs an attitude setpoint $\mathbf{\Psi}_{\text{sp}}$ and a thrust setpoint $\mathbf{T}_{\text{sp}}$. There are two possible choices for the velocity-loop controller and either the MPC controller or the fused-PID (FPID) controller can be used. The MPC controller relies on a model of vehicle dynamics to control the aircraft, which enables it to utilize the full potential of the hybrid vehicle. Due to the computational demand of the numerical optimization involved in MPC, the controller runs on the onboard companion computer. The communication between the companion computer and the autopilot is handled using the robot operating system (ROS).

The FPID outer loop controller runs on the Pixhawk autopilot itself since it needs few computational resources. It was developed in an earlier work and the reader is referred to [24] for details. In the context of this article, it can be seen as a fusion between a multicopter PID velocity controller and a fixed-wing aircraft PID velocity controller. Based on the airspeed of the vehicle, the tilt-angle $\bar{\chi}$ of the propellers is computed. This tilt-angle is then used to determine how the thrust and attitude setpoints generated by the separate multicopter and fixed-wing velocity controllers need to be combined. The FPID controller acts mainly as a fail-safe in case the MPC controller fails to send data to the Pixhawk autopilot (see Section V-E for details).

2) *Attitude Controller:* The velocity-loop controller supplies attitude setpoints $\mathbf{\Psi}_{\text{sp}}$ at a rate of 25 Hz (MPC) or 50 Hz (FPID) to the attitude loop. This inner control loop calculates a torque-triplet $\boldsymbol{\tau}_{\text{sp}}$ to ensure that the attitude of the vehicle follows the attitude setpoint closely. This torque setpoint is then supplied to the control allocation. Internally, the attitude controller is implemented as a cascaded P/PID controller where the outer P-loop calculates an attitude-rate setpoint $\dot{\mathbf{\Psi}}_{\text{sp}}$, which is used as an input to the inner attitude-rate PID control loop. This design allows a high update rate of 200 Hz and a high bandwidth in turn. The gains of the controller need to be tuned, which can, either, be done in simulation and experimentally. For the former approach, the vehicle's known mass and inertia (see Table I) are used to set up a second-order model of the vehicle dynamics. This model can be used subsequently to design and validate the initial controller gains before flight, e.g., by checking Bode plots for well selected inputs/outputs of the attitude loop. Those initial gains are then refined iteratively in experiments to yield satisfactory reference tracking behavior without inducing high-frequency oscillations. Note that tilting the propellers has negligible influence on the inertia of the vehicle as the inertia difference between fixed wing and multicopter configuration is less than 4 % for all axes.

3) *Control Allocation:* The control allocation receives a torque setpoint $\boldsymbol{\tau}_{\text{sp}}$ from the attitude controller and a thrust setpoint $\mathbf{T}_{\text{sp}}$ from the velocity controller. The control allocation maps those high-level commands to actuator outputs. For example, if $\boldsymbol{\tau} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top$ ("pitch up") it could command a positive elevator deflection $\delta_e > 0$.

The velocity loop and attitude controller can be rather generic and the ones presented in this article are suitable for all tilt-rotor VTOL aircraft with wings. This is possible because the exact number of propellers and aerodynamic control surfaces is unknown to the higher control loops. Only the control allocation needs to be specifically tailored to match the vehicle's set of actuators. This also makes it possible to adapt to system failures. Active fault-tolerant control can be achieved without the need to change the velocity or attitude control loop. Only a suitable control allocation that takes the given failure into account [25] is required.

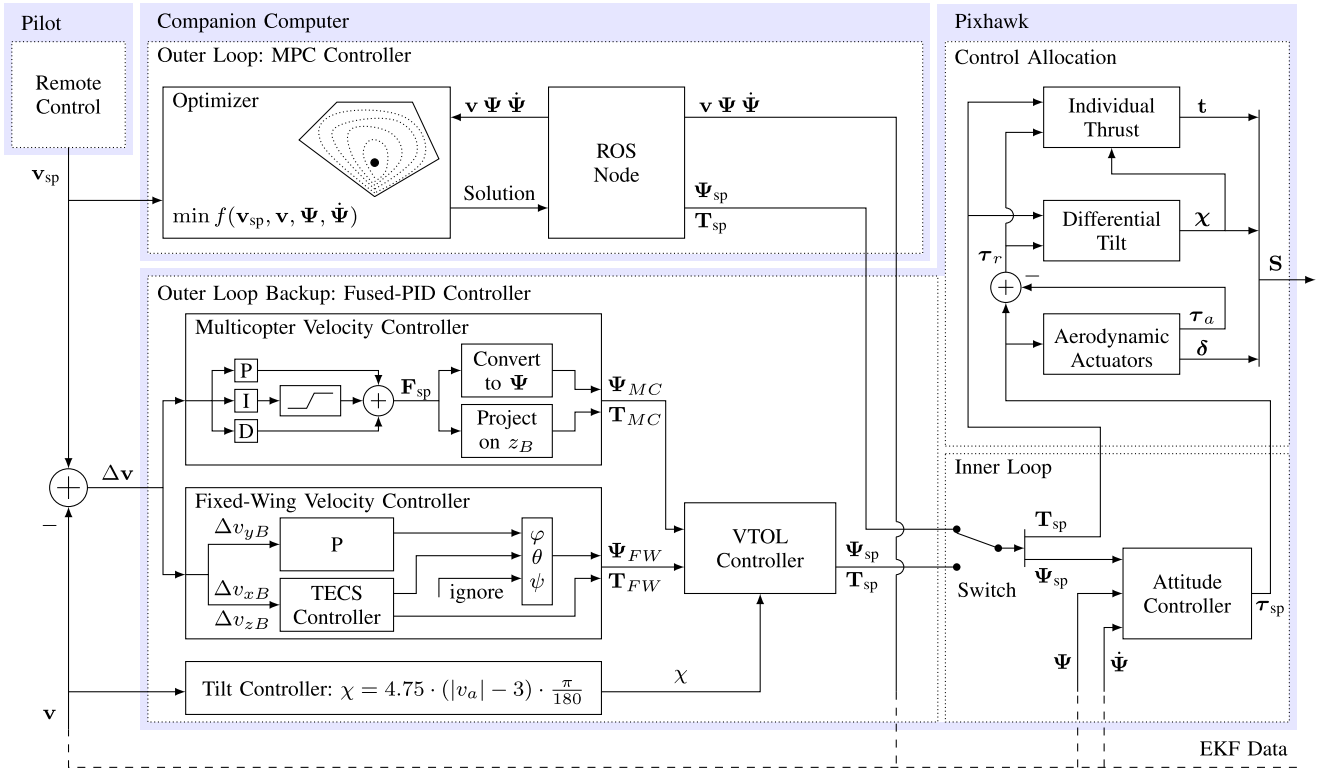

Fig. 3. Control diagram gives an overview over the whole control architecture of the vehicle. The user can choose between using the MPC controller (running on the onboard companion computer) and the Fused-PID controller (running on the Pixhawk) as an outer loop controller that calculates an attitude setpoint Ψ_{sp} and a thrust setpoint $\mathbf{T}_{sp}$ based on the pilot's velocity command. The fast inner attitude control loop then outputs a set of torques $\boldsymbol{\tau}$ to ensure that the vehicle tracks the attitude setpoint. Together with the thrusts, the torques are supplied to the control allocation algorithm to calculate the actuator commands $\mathbf{S}$. All modules run on the Pixhawk except for the MPC which runs on the companion computer.

IV. CONTROL ALLOCATION

The control allocation is a central building block of the presented control system and, for example, determines the output of the MPC controller. First, a model, which describes the total thrust and torque based on the actuator states is required. This model is inverted to obtain the actuator states based on the current desired thrust and torque. In Section IV, the body frame is used for all calculations.

A. Actuator Model

The total thrust $\mathbf{T}$ is modeled as the vector sum of the individual thrusts $\mathbf{t}_i$

$$\mathbf{T} = \begin{bmatrix} T_x \\ 0 \\ T_z \end{bmatrix} = \sum_{i=1}^{2} t_i \begin{bmatrix} \sin(\chi_R) \\ 0 \\ -\cos(\chi_R) \end{bmatrix} + \sum_{i=3}^{4} t_i \begin{bmatrix} \sin(\chi_L) \\ 0 \\ -\cos(\chi_L) \end{bmatrix}. \tag{3}$$

The moments are independently created by the aerodynamic surfaces and the propellers. The chosen model for the rotor moment $\boldsymbol{\tau}_r$ is

$$\boldsymbol{\tau}_r = \sum_{i=1,3} \left[\tilde{C}\,\mathbf{t}_i + d_{ri} \times \mathbf{t}_i\right] + \sum_{i=2,4} \left[-\tilde{C}\,\mathbf{t}_i + d_{ri} \times \mathbf{t}_i\right]$$

where $\tilde{C} = C_Q/C_T$ and C_T and C_Q are the thrust and resisting torque coefficients, respectively. The distances between the center of gravity and the center of the ith propeller is $d_{ri} = d_i + R_{\chi_{L|R}} d_{ei}$. Here, $R_{\chi_{L|R}}$ describes a rotation around the body y-axis by the angle χ_L or χ_R depending on i. The d_is and d_{ei}s are

$$\begin{aligned}
d_1 &= [-l_3 \quad L_0 \quad -h_0]^\top, & d_{e1} &= [-l_1 \quad 0 \quad -h_1]^\top \\
d_2 &= [\ \ l_4 \quad L_0 \quad -h_0]^\top, & d_{e2} &= [\ \ l_1 \quad 0 \quad -h_1]^\top \\
d_3 &= [\ \ l_4 \quad -L_0 \quad -h_0]^\top, & d_{e3} &= [\ \ l_1 \quad 0 \quad -h_1]^\top \\
d_4 &= [-l_3 \quad -L_0 \quad -h_0]^\top, & d_{e4} &= [-l_1 \quad 0 \quad -h_1]^\top.
\end{aligned}$$

The chosen model for the aerodynamic moment is

$$\boldsymbol{\tau}_a = \bar{q}\,S \begin{bmatrix} b\,C_{L,a}\,\delta_a & \bar{c}\,C_{M,e}\,\delta_e & b\,C_{N,r}\,\delta_r \end{bmatrix}^\top \tag{4}$$

where $S, \bar{c}, b$, and $C_{(\cdot)}$ are defined in Table I, $\bar{q}$ is the dynamic pressure and δ_j $(j = a, e, r)$ are the deflections of the aerodynamic actuators. Both rudders are operated in parallel $(\delta_{r,L} = \delta_{r,R} = \delta_r)$ and the two ailerons are operated with the same magnitude but opposite sign $(-\delta_{a_L} = \delta_{a_R} = \delta_a)$.

The total torque $\boldsymbol{\tau}$ generated by the actuators is

$$\boldsymbol{\tau} = \boldsymbol{\tau}_r + \boldsymbol{\tau}_a. \tag{5}$$

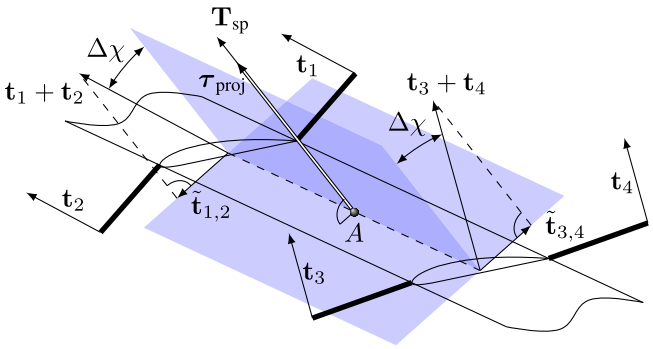

Fig. 4. Control allocation utilizes differential propeller tilting as depicted here. The left and right propellers are tilted independently to produce a thrust matching the setpoint $\mathbf{T}_{\text{sp}}$ and the desired torque along this thrust vector $\boldsymbol{\tau}_{\text{proj}}$ simultaneously. Due to this independent tilting, the total thrusts on the left and right side of the vehicle (i.e. $\mathbf{t}_1 + \mathbf{t}_2$ or $\mathbf{t}_3 + \mathbf{t}_4$) point in different directions. Therefore, the projections of these ($\tilde{\mathbf{t}}_{1,2}$ and $\tilde{\mathbf{t}}_{3,4}$) onto a plane normal to the thrust setpoint are nonzero and point in opposite directions. This generates a torque along the thrust vector, whose magnitude is a function of $\Delta\chi$ and the thrust of the individual propellers.

B. Algorithm

As seen in Fig. 3, the control allocation algorithm converts

$$\mathbf{T}_{\text{sp}} = \begin{bmatrix} T_{\text{sp},x} & 0 & T_{\text{sp},z} \end{bmatrix}^\top, \qquad \boldsymbol{\tau}_{\text{sp}} = \begin{bmatrix} L_{\text{sp}} & M_{\text{sp}} & N_{\text{sp}} \end{bmatrix}^\top$$

to the control vector

$$\mathbf{S} = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 & \chi_L & \chi_R & \delta_a & \delta_e & \delta_r \end{bmatrix}^\top.$$

The control allocation calculates the actuator commands in the following three steps:

1) aerodynamic actuator deflections $\delta_a, \delta_e, \delta_r$;
2) tilt angles χ_L, χ_R;
3) individual propeller thrust t_1, t_2, t_3, t_4.

The aerodynamic actuators can generate torques independently of the propellers as soon as there is airflow over the control surfaces. Generating torques using different propeller speeds requires more energy and, thus, as much of the desired torque as possible is realized with the aerodynamic actuators. Once the aerodynamic torque $\boldsymbol{\tau}_a$ is known, it is subtracted from the setpoint $\boldsymbol{\tau}_{\text{sp}}$ to obtain the residual torque $\boldsymbol{\tau}_r$, which has to be generated by the propellers. A part of this torque can be generated through *differential tilt*, i.e., by choosing a different tilt angle for the left and right propeller pair (see Fig. 4). Calculating the individual tilt angles is done in step 2. In the final step, the individual propeller thrusts are calculated such that the desired overall torque is achieved.

1) *Aerodynamic Actuator Deflections:* The deflections are determined as

$$\begin{aligned} \delta_a &= L_{\text{sp}}/(C_{La}\, S\, b\, \bar{q}) \\ \delta_e &= (M_{\text{sp}} \underbrace{- \frac{l_3 - l_4}{2}\, T_{\text{sp},z} + h_0\, T_{\text{sp},x}}_{\tau_\theta})/(C_{Me}\, S\, \bar{c}\, \bar{q}) \\ \delta_r &= N_{\text{sp}}/(C_{Nr}\, S\, b\, \bar{q}). \end{aligned}$$

This represents an inversion of (4) with an added pitch-torque term τ_θ, which arises because the propellers' thrust is not applied at the center of mass. At low air speeds v_{air} the dynamic pressure $\bar{q} = 0.5\rho_{\text{air}} v_{\text{air}}^2$ tends to zero. To avoid aggressive actuator movements at these low speeds, the deflections are multiplied with a ramp function f_1, which is zero near hover and one at larger air speeds

$$f_1(\bar{q}) = \min(\max(0, a_{r1} \cdot (\bar{q} - b_{r1}) + 0.5), 1).$$

The coefficients a_{r1} and b_{r1} can be found in Table I. Once the deflections are multiplied by this ramp, they are saturated and used to determine the residual torque $\boldsymbol{\tau}_r = \begin{bmatrix} L_r & M_r & N_r \end{bmatrix}^\top$

$$\begin{aligned} L_r &= L_{\text{sp}} - C_{La}\, S\, b\, \bar{q}\, \delta_{a,\text{sat}} \\ M_r &= M_{\text{sp}} - C_{Me}\, S\, \bar{c}\, \bar{q}\, \delta_{e,\text{sat}} \\ N_r &= N_{\text{sp}} - C_{Nr}\, S\, b\, \bar{q}\, \delta_{r,\text{sat}}. \end{aligned}$$

2) *Propeller Tilt Angles:* The propeller tilt angles are chosen such that the thrust set point $\mathbf{T}_{\text{sp}}$ and the residual torque along $\mathbf{T}_{\text{sp}}$ are realized by a symmetric differential tilt, as shown in Fig. 4. The mean tilt angle $\bar{\chi}$ is computed as

$$\bar{\chi} = \arctan\left(\frac{T_{\text{sp},x}}{-T_{\text{sp},z}}\right).$$

The symmetric deviation from the mean tilt angle $\Delta\chi$ is then computed as

$$\Delta\chi = \arctan\left(\frac{\tau_{\text{proj}}\, f_2(||\mathbf{T}_{\text{sp}}||)}{||\mathbf{T}_{\text{sp}}||\, L_0}\right)$$

where $\tau_{\text{proj}} = \boldsymbol{\tau}_r^\top \mathbf{T}_{\text{sp}}/||\mathbf{T}_{\text{sp}}||$ is the projection of the residual torque onto the thrust set point and $f_2(||\mathbf{T}_{\text{sp}}||)$ is a second ramp function, which limits the amount of differential tilt while the thrust is low

$$f_2(||\mathbf{T}_{\text{sp}}||) = \min(\max(0, a_{r2} \cdot (||\mathbf{T}_{\text{sp}}|| - b_{r2})), 1).$$

The coefficients a_{r2} and b_{r2} can be found in Table I. Now, the tilt angles χ_L and χ_R can be computed as

$$\chi_L = \bar{\chi} - \Delta\chi \tag{6}$$
$$\chi_R = \bar{\chi} + \Delta\chi\,. \tag{7}$$

Finally, $\Delta\chi$ is reduced such that both tilt angles χ_L and χ_R are within the valid range (-7° to 90°). This guarantees that the total thrust points in the direction commanded by the velocity controller.

3) *Individual Propeller Thrust:* Once the tilt angles are known, the thrust and torque model of the propellers becomes linear in the individual thrust components

$$\begin{bmatrix} T_{\text{sp},x} \\ T_{\text{sp},z} \\ L_r \\ M_r \\ N_r \end{bmatrix} = \begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix} \tag{8}$$

with

$$\mathbf{A} = \begin{bmatrix} \sin(\chi_R) & \sin(\chi_R) \\ -\cos(\chi_R) & -\cos(\chi_R) \\ -L_0\cos(\chi_R)+\frac{C_Q}{C_T}\sin(\chi_R) & -L_0\cos(\chi_R)-\frac{C_Q}{C_T}\sin(\chi_R) \\ -l_1-l_3\cos(\chi_R)-h_0\sin(\chi_R) & l_1+l_4\cos(\chi_R)-h_0\sin(\chi_R) \\ -L_0\sin(\chi_R)-\frac{C_Q}{C_T}\cos(\chi_R) & -L_0\sin(\chi_R)+\frac{C_Q}{C_T}\cos(\chi_R) \end{bmatrix}$$

$$\begin{bmatrix} \sin(\chi_L) & \sin(\chi_L) \\ -\cos(\chi_L) & -\cos(\chi_L) \\ L_0\cos(\chi_L)+\frac{C_Q}{C_T}\sin(\chi_L) & L_0\cos(\chi_L)-\frac{C_Q}{C_T}\sin(\chi_L) \\ l_1+l_4\cos(\chi_L)-h_0\sin(\chi_L) & -l_1-l_3\cos(\chi_L)-h_0\sin(\chi_L) \\ L_0\sin(\chi_L)-\frac{C_Q}{C_T}\cos(\chi_L) & L_0\sin(\chi_L)+\frac{C_Q}{C_T}\cos(\chi_L) \end{bmatrix}.$$

In this last step of the algorithm, (8) is solved to obtain the individual thrusts t_i.

C. Optimality

The proposed algorithm is an approximation of the optimal mapping of thrust and torque to the control vector. To show the effectiveness of the differential tilt and the performance of the algorithm in general, the following three control allocations are compared:

1) control allocation based on nonlinear optimization;
2) the proposed algorithm;
3) the proposed algorithm without differential tilt.

To compare these, a variety of sample thrust and torque setpoints are sent to all of these algorithms and their outputs are compared (see Fig. 5).

The control allocation algorithm based on nonlinear optimization minimizes the sum of squared thrusts

$$J_{\mathrm{CA}} = t_1^2 + t_2^2 + t_3^2 + t_4^2$$

subject to the actuator model described in Section IV-A. For the control allocation without differential tilt, the second step of the described algorithm is replaced by setting $\Delta\chi$ to zero. Fig. 5 shows the total cost J_{CA} produced by all three algorithms and compares the propeller tilt angles of the proposed algorithm with the optimal ones. Without the differential tilt, the cost is much higher compared to the optimal cost. When the differential tilt is enabled, the cost is almost identical to the optimal cost. However, the computation time required to solve the optimization problem is much higher compared to the developed algorithm.

A rare corner-case where the approximation deviates minimally from the optimal solution are maneuvers where both propellers are at a tilt limit (i.e., $-7°$ or $90°$) and a very large roll and yaw torque is demanded simultaneously. In these cases it might be suboptimal to tilt the propellers symmetrically. However, such a scenario is an artificial example as it is highly unlikely to occur during a real flight.

V. MODEL PREDICTIVE CONTROL

This section presents the velocity MPC controller, which takes a velocity setpoint from the pilot as an input. The goal of predictive control, is to use a mathematical model of the vehicle dynamics in order to achieve optimal control performance. The MPC controller computes an optimal

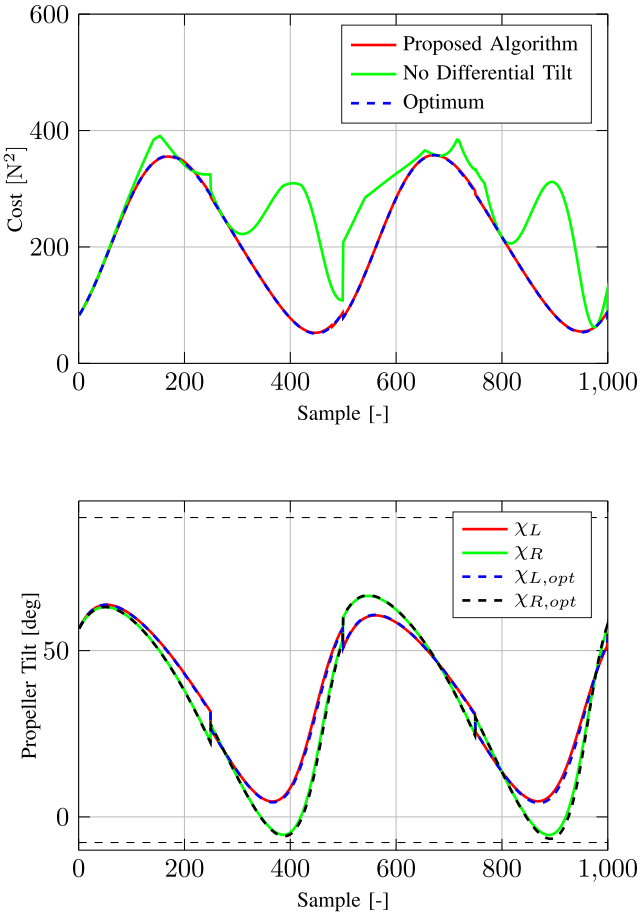

Fig. 5. Performance measure, the described algorithm is compared to the algorithm without differential tilt and the optimal solution. The top diagram shows the resulting cost of all three algorithms. The bottom graph compares the calculated tilt angles of the proposed algorithm with the optimal tilt angles.

sequence $\mathbf{U}$ of inputs $\mathbf{u}_k^\star$ that minimize a cost function $J(\mathbf{x}, \mathbf{u})$ over a horizon of N time steps. The solution respects constraints on the state vector $\mathbf{x}$ and the input vector $\mathbf{u}$

$$\begin{aligned} \mathbf{U} = \arg\min_{\mathbf{u}_k} \quad & \sum_{k=0}^{N} J\left(\mathbf{x}_k, \mathbf{u}_k\right) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{x}_k \in \mathcal{X} \\ & \mathbf{u}_k \in \mathcal{U}(\mathbf{x}_k). \end{aligned} \tag{9}$$

The state vector $\mathbf{x}$ will be discussed along with the input vector $\mathbf{u}$ in the following section. Then, the constraints, objective function $J(\cdot)$ and dynamic model $f(\cdot)$ are explained. Finally, an efficient implementation on the companion computer is presented.

A. Input and State Vectors

Choosing a suitable input $\mathbf{u}$ and state $\mathbf{x}$ is a crucial part of the controller design. Typically, one chooses physically meaningful quantities and hence a straightforward possibility would be to use the velocity, the attitude and the attitude rate as well as the tilt angle as a part of the state. The input

would consequently be the thrust magnitude T and change in tilt angle $\dot{\chi}$ together with the torque triplet

$$\begin{aligned}\boldsymbol{\tau} &= \begin{bmatrix} L & M & N \end{bmatrix}^\top \\ \mathbf{x} &= \begin{bmatrix} \mathbf{v} & \boldsymbol{\Psi} & \dot{\boldsymbol{\Psi}} & \bar{\chi} \end{bmatrix}^\top \in \mathbb{R}^{10\times 1} \\ \mathbf{u} &= \begin{bmatrix} T & \dot{\bar{\chi}} & L & M & N \end{bmatrix}^\top \in \mathbb{R}^{5\times 1}. \end{aligned} \tag{10}$$

This choice would allow the controller to optimize the torques and thrust that are required to control the aircraft to yield good performance. Note that choosing the propeller-tilt angle as a state and using the tilt rate as an input simplifies constraining $\dot{\bar{\chi}}$, which corresponds to the physically limited motor speed of the actuator servo. The velocity is given in the NED-frame as this makes it easier to use a global trajectory planner.

However, this choice of $\mathbf{x}$ and $\mathbf{u}$ is not very well suited to the problem at hand since the MPC controller does not directly send torque setpoints to the control allocation, but instead needs to supply attitude setpoints to the inner loop. Directly sending torques is not possible due to the too slow rate of the MPC (25 Hz) and the time delay introduced by the communication between the companion computer and the autopilot. A possible way to compensate the delay would be to use the computed attitude trajectory as a setpoint and, thus, assume that the attitude controller perfectly tracks this reference. In a real application, however, this assumption is invalid. Consequently, the torques applied to the system by the inner attitude controller do not match the MPC's trajectory, which degrades the control performance.

By incorporating the inner attitude controller into the MPC formulation, the above issue can be resolved. In effect, this inverts the inner attitude controller such that the torques required by the MPC are sent to the control allocation. This can be thought of as increasing the effective rate of the MPC since the inner control loop operates at 200 Hz. Based on these insights, the vector $\mathbf{u}$ is chosen to match the input to the inner control loop (see Fig. 3), i.e., the elements are the thrust T, propeller-tilt rate $\dot{\bar{\chi}}$, and attitude solution $\boldsymbol{\Psi}_{\text{MPC}}$

$$\mathbf{u} = \begin{bmatrix} T & \dot{\bar{\chi}} & \boldsymbol{\Psi}_{\text{MPC}} \end{bmatrix}^\top \in \mathbb{R}^{5\times 1}. \tag{11}$$

The yaw-angle setpoint ψ_{MPC} would be unconstrained but the vehicle's yaw dynamics are 2π-periodic, which can be a problem for the optimizer. Thus, the yaw attitude setpoint is expressed relative to the current yaw of the vehicle and the following equation converts the solution of the MPC into the actual attitude setpoint:

$$\boldsymbol{\Psi}_{\text{sp}} = \boldsymbol{\Psi}_{\text{MPC}} + \begin{bmatrix} 0 & 0 & \psi \end{bmatrix}^\top. \tag{12}$$

The state needs to be expanded by the previous (e.g., from the last iteration of the MPC controller) attitude rates $\dot{\boldsymbol{\Psi}}^-$ to include the P/PID attitude control loop. The dynamics of the integral part are slow and thus neglected in the state. Furthermore, a smooth thrust curve is desirable and therefore the last thrust setpoint T^- is added to the state. This allows penalizing large thrust changes in the objective function. The state vector is then given by

$$\mathbf{x} = \begin{bmatrix} \mathbf{v} & \boldsymbol{\Psi} & \dot{\boldsymbol{\Psi}} & \bar{\chi} & \dot{\boldsymbol{\Psi}}^- & T^- \end{bmatrix}^\top \in \mathbb{R}^{14\times 1}. \tag{13}$$

B. Constraints

The main advantage of model predictive control over other types of controllers is that MPC allows to incorporate constraints. Typically, such constraints represent either the physical limits of the plant that is controlled or safety limitations. In contrast to these *hard constraints*, one can also specify *soft constraints*, which are not strict limits but incur high cost when "violated". The chosen hard constraints for the considered vehicle are given below, where $|\cdot|$ denotes the elementwise absolute value

$$\begin{aligned} |\mathbf{v}| &\leq \begin{bmatrix} 35 & 35 & 10 \end{bmatrix}^\top \text{ms}^{-1} & T &\in [0 \quad 40] \quad N \\ |\boldsymbol{\Psi}| &\leq \begin{bmatrix} \frac{\pi}{4} & \frac{\pi}{4} & \infty \end{bmatrix}^\top & T^- &\in [0 \quad 40] \quad N \\ |\dot{\boldsymbol{\Psi}}| &\leq \begin{bmatrix} \pi & \pi & \pi \end{bmatrix}^\top \text{degs}^{-1} & \bar{\chi} &\in [-7 \quad 90] \text{ deg} \\ |\boldsymbol{\Psi}_{\text{MPC}}| &\leq \begin{bmatrix} \frac{\pi}{3} & \frac{\pi}{3} & \frac{\pi}{2} \end{bmatrix}^\top & \dot{\bar{\chi}} &\in [-45 \quad 45] \text{degs}^{-1}. \end{aligned} \tag{14}$$

C. Objective Function

The most important part of any predictive controller is the objective function that it minimizes. The objective determines, which control actions and flying states are desirable (thus, "cheap"), how well the reference will be tracked and which flying maneuvers are better avoided. To achieve the latter, soft constraints J_{soft} incurring large cost when violated are incorporated into the objective function. To ensure reference tracking, the objective includes a cost J_{ref} that grows as the tracking error increases. To avoid excessive actuator usage and overly aggressive flying maneuvers (e.g., very high attitude rates) a cost term $J_{\text{x,u}}$ that depends on the state and input is introduced. Overall, the objective function has the following form:

$$J(\mathbf{x}_k, \mathbf{u}_k) = J_{\text{ref}}(\mathbf{v}, \mathbf{v}_{\text{sp}}) + J_{\text{x,u}}(\mathbf{x}_k, \mathbf{u}_k) + J_{\text{soft}}(\mathbf{x}_k).$$

1) *Reference Tracking:* A typical cost function for reference tracking is a squared-error objective with a weight matrix $Q_{\text{ref}} = \text{diag}(q_x, q_y, q_z)$, i.e., $J_{\text{ref}} = (\mathbf{v} - \mathbf{v}_{\text{sp}})^\top Q_{\text{ref}}(\mathbf{v} - \mathbf{v}_{\text{sp}})$. The shortcoming of this approach is that the vehicle will not maintain its altitude in order to reduce the velocity error cost. Consider the following situation: the vehicle is at rest and the velocity setpoint changes to 20 m/s. The objective function will then attain the value $400q_x$. However, by descending the vehicle can accelerate much faster while only incurring a minor cost (e.g., $v_z = 2\,m/s$ with cost $4q_z$). Therefore, the vehicle will not maintain its altitude.

To resolve this problem, an ℓ_1 objective is used instead of the ℓ_2 objective

$$J_{\text{ref}}(\mathbf{v}, \mathbf{v}_{\text{sp}}) = ||\mathbf{q}_{\text{ref}}^\top \times \left(\mathbf{v} - \mathbf{v}_{\text{sp}}\right)_\psi ||_1. \tag{15}$$

TABLE II
Weights for the State and Input Cost

$\mathbf{Q}_\Psi$			$\mathbf{Q}_{\dot\Psi}$			$\mathbf{Q}_\mathrm{u}$					q_t
ϕ	θ	ψ	$\dot\phi$	$\dot\theta$	$\dot\psi$	T	$\dot\chi$	ϕ_sp	θ_sp	ψ_sp	
20	20	0	5	5	0	0.0025	1	100	200	50	40

The vector of weights $\mathbf{q}_\mathrm{ref} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\top = \begin{bmatrix} 5 & 5 & 10 \end{bmatrix}^\top$ determines how "costly" a deviation from the velocity setpoint is. The safety-critical z direction gets more weight than the x and y direction. Minimizing the ℓ_1 norm is known to reduce the cardinality [26], which makes a NED coordinate frame unsuitable. The vehicle would first try to minimize the velocity error along either of the coordinate axes and, therefore, turn exactly north, east, etc. Choosing the body frame to perform the calculation is equally unsuitable as the value of J_ref would then depend on the pitch and roll of the vehicle due to the larger cost in the z direction. Therefore, the velocity error is calculated in a coordinate frame, which is only rotated by the yaw angle ψ but not the other two Euler angles corresponding to roll and pitch. In this coordinate frame, reducing the cardinality corresponds to holding the altitude and the course.

To improve the performance of the numerical solver, a soft, continuously differentiable approximation $|x|_\mathrm{s}$ of the absolute value function is used in the ℓ_1 norm

$$|x|_\mathrm{s} = 2\alpha \log(1 + \exp(x/\alpha) - x - 2\alpha \log(2)\,.$$

The parameter $\alpha = 0.1$ controls the softness of the approximation and needs to be tuned empirically to yield good numerical performance.

2) *State and Input Cost:* The state and input costs are given by quadratic functions in each element of the state and input vector

$$\begin{aligned} J_\mathrm{x,u}(\mathbf{x}_k, \mathbf{u}_k) = {} & \boldsymbol{\Psi}^\top \mathbf{Q}_\Psi \boldsymbol{\Psi} + \dot{\boldsymbol{\Psi}}^\top \mathbf{Q}_{\dot\Psi} \dot{\boldsymbol{\Psi}} \\ & + \mathbf{u}_k^\top \mathbf{Q}_\mathrm{u} \mathbf{u}_k + q_t\,(T^- - T)^2 \end{aligned} \tag{16}$$

where the matrices $\mathbf{Q}$ are diagonal matrices with entries according to Table II. Initially, the weights have been chosen such that all states and inputs have a cost of about one during typical operating conditions of the vehicle. To improve the performance of the controller, the weights have been empirically tuned during the testing process.

3) *Soft Constraints:* The use of soft constraints can either be interpreted as a "suggestion" to the controller or as an actual constraint that does not render the optimization problem infeasible when it is violated. Soft constraints are used to recommend a sensible tilt angle and to constrain the velocity.

For safety reasons, it is undesirable to have the propellers tilted forwards at low speeds, but there is no strict limit that needs to be enforced. At high speeds all configurations are allowed. Hence, a soft constraint $J_{\mathrm{soft},\bar\chi}$ that encodes the "recommendation" to keep the propellers pointed upwards at low forward speeds $v_{x,B}$ is used

$$\begin{aligned} J_{\mathrm{soft},\bar\chi}(v_{x,B}, \bar\chi) &= q_{\bar\chi} \exp(a \cdot v_{x,B} \cdot \bar\chi + b \cdot \bar\chi + c \cdot v_{x,B} + d) \\ a &= -0.332,\, b = 13.35,\, c = -0.477,\, d = -2.303. \end{aligned}$$

For a more intuitive understanding, some values of this function are given below in Table III.

TABLE III
Some Values of the Cost Function

$\bar\chi$ \ v	0 m/s	5 m/s	10 m/s	15 m/s	20 m/s	25 m/s
0 deg	0.1	9.2e-3	8.5e-4	7.8e-5	7.2e-6	6.6e-7
45 deg	3.5e3	89.4	2.24	0.056	1.4e-3	3.5e-5
90 deg	1.2e8	8.7e5	5.9e3	40	0.27	1.8e-3

Soft constraints are also applied to the velocity. The box constraints given in (14) act on the velocity of the vehicle in the inertial frame, and thus, the aerodynamic limitations of the vehicle are not modeled. To resolve this, a soft constraint $J_{\mathrm{soft},v}$ that depends on the velocity $\mathbf{v}_B$ (in the body frame) is introduced. The aerodynamic characteristics of the vehicle do not allow it to fly sideways at high speeds and also do not permit fast backward-flying. Therefore, the applied soft constraints are

$$v_{x,B} > -1\ \mathrm{ms}^{-1}, \qquad |v_{y,B}| \le 2\ \mathrm{ms}^{-1}, \qquad |v_{z,B}| \le 2\ \mathrm{ms}^{-1}.$$

They are implemented using a weighted sum of exponential functions as follows:

$$\begin{aligned} J_{\mathrm{soft},v} = {} & q_x \left(\exp\left(-3(v_{x,B}+1)\right) + k_x v_{x,B} - c_x\right) \\ & + q_y \left(\exp\left(-v_{y,B}-2\right) + \exp\left(v_{y,B}-2\right) - c_y\right) \\ & + q_z \left(\exp\left(-v_{z,B}-2\right) + \exp\left(v_{z,B}-2\right) - c_z\right). \end{aligned}$$

The coefficient k_x and the offsets $c_{[x,y,z]}$ are chosen such that the cost function and its derivative are zero when the vehicle is at rest. Otherwise, the vehicle would move in some direction even if the pilot does not input such a command.

D. Vehicle Dynamics

The dynamics model in the MPC framework is used to predict how the state of the system will evolve when a series of control inputs is applied to it. The aerodynamic model and the model of the PID attitude controller are explained below.

1) *Aerodynamic Model:* Accurate aerodynamic models are quite complex and require a lot of computing power to evaluate, which makes them unsuited for an application in a real-time controller onboard the vehicle. Instead, a simplified model that only accounts for the most important dynamics has been implemented. The surfaces [i.e., wing (W), horizontal (H) and vertical (V) stabilizer, and fuselage (F)] need to be taken into account here. The model assumes that a surface generates a drag force $\mathbf{F}_D^{[W,H,V,F]}$ opposite to the incoming airflow. A wing or flat plate (stabilizers) also generates a lift force $\mathbf{F}_L^{[W,H,V]}$ perpendicular to the direction of the airflow. The fuselage is assumed to produce no lift.

The lift and drag force can, then, be calculated as

$$\mathbf{F}_{L|D} = \frac{1}{2}\rho v_{\perp}^2 A c_{l|d}(\alpha) \cdot \mathbf{e}_{l|d} \tag{17}$$

where ρ is the air density, A the surface area of the component, c_l the coefficient of lift, c_d the coefficient of drag, $v_\perp$ the airspeed perpendicular to the front edge of the wing, and $\mathbf{e}_{l|d}$ the unit vector in the direction of the lift or drag. Note that the angle of attack α in (17) for the vertical stabilizer is the side-slip angle β and not the usual angle between the chord of a horizontal wing and the air stream.

The coefficients of lift and drag for the main wing of the aircraft are given in [23] and a continuously differentiable approximation of $c_l^W(\alpha)$ and $c_d^W(\alpha)$ is provided as well. The MPC uses this approximation with modified parameters for improved numerical performance. It is reproduced below for the sake of completeness

$$\begin{aligned}
\sigma(\alpha) &= (1 - \tanh(-1.030 + 20\alpha^2)) \\
c_d^W(\alpha) &= 0.5637\sigma(\alpha) \cdot (0.03 + 0.2\alpha^2) \\
&\quad + (1 - 0.5637\sigma(\alpha)) \cdot (0.025 + 2\sin(\alpha)^2) \\
c_l^W(\alpha) &= 0.5637\sigma(\alpha) \cdot (0.25 + 5.62\alpha) \\
&\quad + (1 - 0.5637\sigma(\alpha) \cdot \sin(2\alpha).
\end{aligned}$$

The horizontal and the vertical stabilizers of the aircraft are assumed to be flat plates, whose prestall lift and drag can be roughly approximated by [27]

$$c_d^{[H,V]}(\alpha) = 0.8625|\alpha|, \qquad c_l^{[H,V]}(\alpha) = 0.885\alpha.$$

The fuselage of the aircraft is approximated as a square beam with an aspect ratio of 4.35, which only produces drag during side slipping. The drag coefficient $c_d^F = 1.28$ is obtained by interpolating the values given for different aspect ratios in [28]. The surface areas of the aircraft's components are given in Table I.

The aerodynamic model also includes the torques $\boldsymbol{\tau}_{\text{aero}}$ caused by the lift and drag forces using the cross product between the force and the vector $\mathbf{r}$ from the center of gravity to the center of lift of the respective component/wing

$$\boldsymbol{\tau}_{LD} = \mathbf{r} \times (\mathbf{F}_L + \mathbf{F}_D).$$

The total force $\mathbf{F}_{\text{aero}}$ and torque $\boldsymbol{\tau}_{\text{aero}}$ (in the body frame) are then given by the sum of the forces $\mathbf{F}_{L|D}$ and torques $\boldsymbol{\tau}_{LD}$ over all components.

2) *P/PD Attitude Controller:* As shown in Fig. 3, the MPC supplies attitude setpoints to the inner-loop controller. Thus, this attitude controller needs to be incorporated in the dynamic model as well. A P controller first computes an attitude rate setpoint $\dot{\boldsymbol{\Psi}}_{\text{sp}}$ based on the current attitude and the setpoint given to it

$$\dot{\boldsymbol{\Psi}}_{\text{sp}} = \text{diag}\left(k_{p,\text{roll}}, k_{p,\text{pitch}}, k_{p,\text{yaw}}\right) \times \left(\boldsymbol{\Psi}_{\text{sp}} - \boldsymbol{\Psi}\right).$$

Based on this setpoint a PD controller calculates a torque that will be generated by the vehicle

$$\begin{aligned}
\boldsymbol{\tau} &= \text{diag}(k_{p,\text{rollrate}}, k_{p,\text{pitchrate}}, k_{p,\text{yawrate}}) \times (\dot{\boldsymbol{\Psi}}_{\text{sp}} - \dot{\boldsymbol{\Psi}}) \\
&\quad + \text{diag}(k_{d,\text{rollrate}}, k_{d,\text{pitchrate}}, k_{d,\text{yawrate}}) \times (\dot{\boldsymbol{\Psi}}^{-} - \dot{\boldsymbol{\Psi}}).
\end{aligned}$$

3) *State-Update Equation:* With known aerodynamic forces and torques as well as the attitude controller, the state-update equation from (9) can be written in continuous time as

$$\dot{\mathbf{x}} = \begin{bmatrix} \frac{1}{m} \cdot {}_G\mathbf{R}_B(\mathbf{T} + \mathbf{F}_{\text{aero}}) + 9.81\mathbf{e}_z \\ \dot{\boldsymbol{\Psi}} \\ \mathbf{I}^{-1}(\boldsymbol{\tau} + \boldsymbol{\tau}_{\text{aero}}) \\ \dot{\bar{\chi}} \end{bmatrix} \tag{18}$$

where m is the mass of the vehicle and the inverse its inertia matrix $\mathbf{I}^{-1} = \text{diag}(11.236, 14.925, 8)$. The states $\dot{\boldsymbol{\Psi}}^{-}$ and T^{-} are simply copied from the last iteration and (18) is numerically integrated using a fourth order Runge–Kutta scheme to arrive at the discrete-time formulation.

E. Implementation

The MPC implementation relies on the commercial *FORCES Pro* solver by *Embotech* [29], [30], which is used because of its ability to solve nonlinear MPC problems quickly. For the presented MPC architecture, the average runtime is 33 ms with a standard deviation of 5 ms on the 4×1.44 GHz Intel UpBoard CPU. The problem is coded in MATLAB and the software then outputs an optimized solver as a C object file. This solver is then interfaced through the ROS with a custom *node*, as shown in Fig. 3: the Pixhawk autopilot's inner-loop constantly publishes the state of the aircraft. The ROS node receives the data and sends the current state and user command to the MPC solver. The results of the MPC solver are then sent back to the Pixhawk. On the MPC side, the communication and computation are running on different threads to avoid interference. The communication between the UpBoard and the PixHawk is done using MAVLink, which is a very lightweight serial protocol. Internally, the Pixhawk uses asynchronous communication to minimize the computational load of the communication. Overall, the computation has no measurable impact on the system performance.

The main difficulty is that the optimizer takes a varying amount of time to solve the problem whereas the autopilot needs to receive new setpoints periodically. This is achieved with a two-thread architecture where the main thread handles the communication and the second thread interfaces and runs the optimizer. If the optimizer finishes in less than 40 ms, the newly computed solution is sent to the attitude controller. If the optimizer does not finish, the next iterate from the last solution is used (case a). This is possible because the MPC always computes an optimal solution over $N = 20$ time steps. Only if the optimizer finds no solution after 20 time steps (equivalent to 0.8 s), the Pixhawk automatically switches to the backup FPID controller (case b), as shown in Fig. 3. In about 9.1 % of the iterations, the MPC did not finish within 40 ms (case a), but the emergency switching to the backup controller (case b) never happened.

At each time step, the MPC controller with horizon length N computes a trajectory of length N. If the optimizer does not find a solution fast enough, the trajectory from the last solution is used to send setpoints to the Pixhawk.

However, after N steps this is not possible anymore and the system would automatically switch to the backup FPID controller. Note that this has not been observed in any experiment.

The two-thread architecture also makes it possible for the second thread to efficiently manage the memory required by the optimizer. By only allocating the memory once and reusing it fully in each iteration, memory bandwidth is saved. Furthermore, proper memory alignment is guaranteed when allocating the memory in the beginning.

VI. EXPERIMENTAL RESULTS

In order to verify the performance of the MPC controller, experiments in simulation and with a remote-controlled VTOL airplane have been performed. The simulator was especially useful to test maneuvers that would have been very difficult in an outdoor environment due to space constraints and official UAV regulations. First the simulation setup is described and a flight to validate the simulation is presented. Then, the results from simulation and outdoor experiments are shown. For the outdoor flights, localization is done by fusing the inertial navigation unit data (accelerometer, gyroscope, magnetometer) with GPS and airspeed measurements from a pitot tube with an extended Kalman filter.

A. Simulation

1) *Environment:* The firmware of the Pixhawk autopilot is equipped with a software-in-the-loop feature that can be interfaced with the *Gazebo*[1] simulator for the dynamics simulation and a visualization. The MPC controller runs, like on the real vehicle, on the companion computer, which communicates with the simulated Pixhawk via a network connection. Apart from the dynamics simulation done by Gazebo, the simulator is identical to the real vehicle (identical code, only compiled for x86 architecture), which greatly facilitates developing and testing. The simulation has been tuned based on numerous experiments in order to closely match the actual aircraft's behavior.

2) *Verification:* To be able to use the simulator not only for testing, but also as a substitute to actual outdoor experiments, its accuracy needs to be validated. This was done by conducting an identical flight in simulation and with the remote-controlled vehicle. The flight profile chosen for this task is a velocity step from 0-20 m/s because this covers a large velocity range and all of the flight configurations between, and including, the multicopter and fixed-wing modes are contained. The deceleration could not be part of this comparison, as a steep climb had to be flown manually to stop within the available range and line of sight.

The results of the test are shown in Fig. 6, where the dashed lines correspond to the simulation and the solid lines represent the real-world experiment. As soon as the velocity reference jumps to 20 m/s, the airplane accelerates as quickly as possible and reaches the velocity target after 8.5 s while maintaining close-to-zero vertical velocity. One can see that the two flight profiles match within less than 1.5 m/s, which proves the accuracy of the simulation. The outdoor flight has a slightly noisier vertical velocity v_z, which is most likely due to the presence of a light wind[2] (1 m/s, direction 280° w.r.t. north) and the turbulence induced by the propellers.

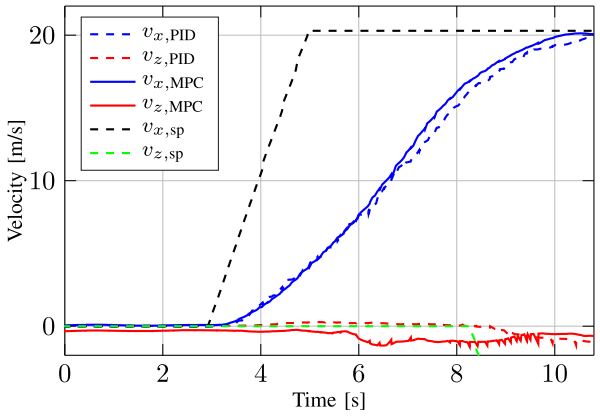

Fig. 6. Plot shows a comparison between the actual aircraft flight (solid lines) and the simulation flight (dashed lines). The horizontal velocity setpoint is shown in black, the vertical velocity setpoint in green. The experiment shows that the simulation flight matches the remote-controlled aircraft flight very well.

3) *MPC Versus FPID Controller:* In a previous work, the FPID control architecture was developed [24], which is used in this article. The most prominent shortcoming of the FPID controller is its inability to slow down the vehicle at high speeds while flying approximately level (i.e., not climbing or descending). The MPC, on the other hand, should be able to track the reference more closely, because it can fully utilize the nonlinear dynamics of the VTOL aircraft. To compare both controllers, a ramp with an acceleration of 2 m/s^2 to a velocity of 20 m/s and subsequent deceleration is tested in simulation. The results are summarized in Fig. 7. The curves corresponding to the MPC are drawn with solid lines, whereas the FPID is shown in dashed lines.

When the vehicle is asked to accelerate, the MPC controller does so without having to pitch down because it tilts the propellers forward. This allows it to immediately begin tracking the reference without descending. The vehicle continues with nearly constant acceleration until it slightly overshoots 20 m/s. The FPID controller on the other hand is unable to use the tilting mechanism to its advantage and pitches down to gain forward momentum. It eventually overshoots the ramp and is yet unable to reach the target speed. However, both controllers track the vertical velocity reference $v_{z,\text{sp}} = 0$ m/s very well.

[1] [Online]. Available: http://gazebosim.org/

[2] [Online]. Available: https://kachelmannwetter.com/de/messwerte/loerrach/windmittel-10 min/20 200 709-1830z.html

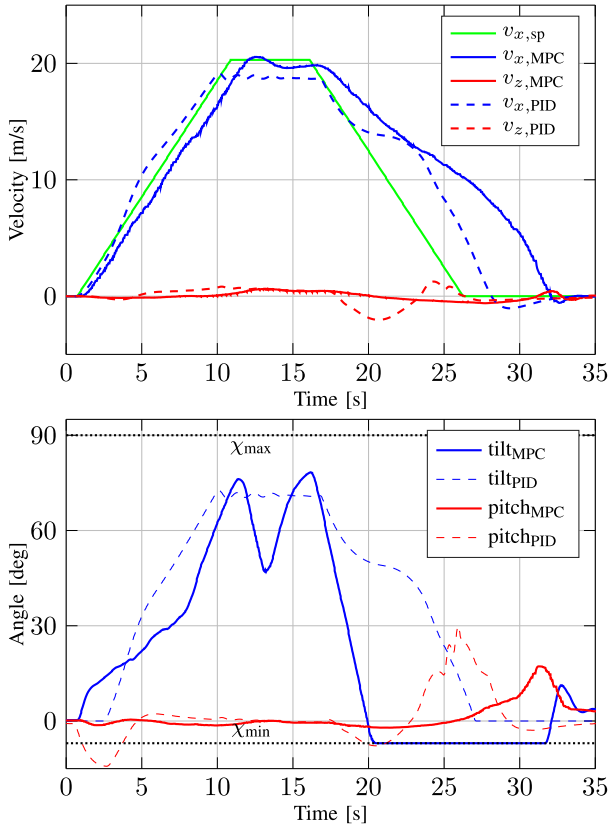

Fig. 7. Comparison between the FPID (dashed lines) and MPC controller (solid lines): the top plot shows the velocity tracking capabilities of the controllers and the bottom plot compares the pitch and tilt angles of the vehicle during the same flight. The MPC uses the tilting mechanism fully to accelerate and decelerate more smoothly. The tilt-angle limits are indicated with dotted lines.

The advantage of the MPC becomes more obvious during the deceleration phase of the flight: initially the FPID perfectly tracks the ramp, but is only able to do so by rapidly climbing. There is not enough drag to slow the vehicle down quickly enough and thus the only option is to trade vertical velocity tracking error for horizontal velocity tracking error. The MPC on the other hand tilts its propellers back fully ($-7°$). After $t = 25\,s$ this becomes an advantage as the backwards tilted propellers can, now, be used to gradually compensate the decreasing lift of the main wing while pointing back slightly and thus improving deceleration. The FPID controller needs to pitch up much more aggressively during the deceleration and is not able to maintain close-to-zero vertical velocity.

This simulated experiment shows that the MPC controller indeed has an advantage over the FPID controller. It is able to track the reference more closely during the acceleration and is able to decelerate much more smoothly. Being able to slow down without flying the aircraft at poststall angles of attack is an important safety feature.

B. Outdoor Flights

In order to properly demonstrate the MPC controller's capabilities, outdoor flights with the tilt-rotor VTOL aircraft

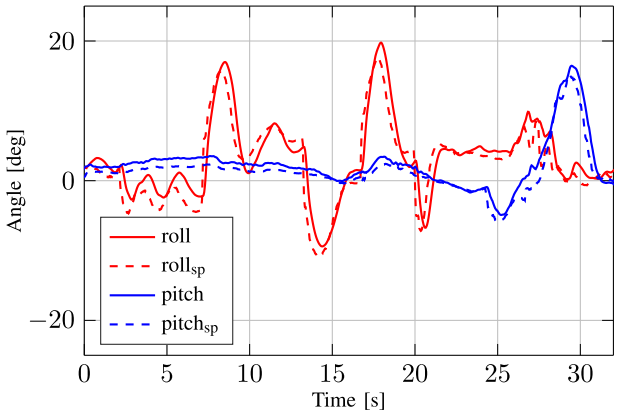

Fig. 8. Plot shows the control performance of the inner loop: it can track the roll and pitch reference accurately even when larger angles are demanded.

are presented. The pilot uses the remote control to send velocity setpoints only and is not able to "manually" fly the aircraft unless an emergency-override switch is activated first. All pilot velocity commands are clearly indicated in the plots below, e.g., $v_{x,\mathrm{sp}}$ and $v_{z,\mathrm{sp}}$. First, a flight showing the inner-loop attitude control performance is shown and then two flights with the MPC controller are presented: a hover test where no wind (<1 bft) was present and an acceleration/deceleration flight in 1 bft ($\approx$ 1 m/s) average wind and 2 bft ($\approx$ 2.5 m/s) gust conditions. A short video montage of the flights is available at https://youtu.be/LhqsPZrddw4

1) *Attitude Controller:* The inner-loop attitude controller is a P/PID controller that needs to be tuned to properly track the attitude reference computed by the MPC controller. Fig. 8 shows the roll and pitch reference tracking while flying maneuvers in the xy plane at low speeds. Despite the large wings that damp the roll of the vehicle, the controller is able to track the reference with minimal delay. The pitch tracking is very similar, but there is a steady offset between the reference and the actual pitch. This is because the inner-loop controller of the Pixhawk autopilot does not feature an integrator in the attitude loop, but only in the attitude-rate loop. An offset center of gravity (e.g., due to battery placement), thus, leads to the observed behavior.

2) *Hover:* To verify the controller's ability to reject disturbances, a hover test with a zero-velocity setpoint was conducted. Fig. 9 shows the velocities along all three axes of the body frame. They are within 0.1 m/s of their reference. This shows that the predictive controller is able to stabilize the vehicle and hover accurately.

3) *Acceleration/Deceleration:* The test flight that was used to validate the simulation already shows the controller's ability to accelerate the vehicle to high velocities. To also show that the controller can utilize intermediate flight configurations, another flight with a lower top speed was conducted. The result is shown in Fig. 10. The measured values are represented by solid lines whereas setpoints are plotted with dashed lines.

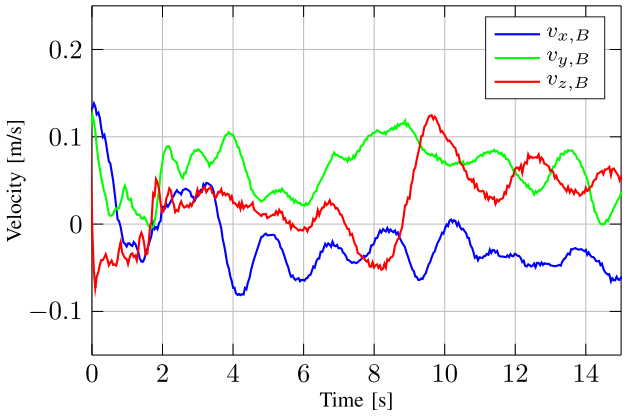

Fig. 9. Velocity reference was set to zero for all three axes. The MPC controller tracks this reference very well under low wind conditions (<1 bft).

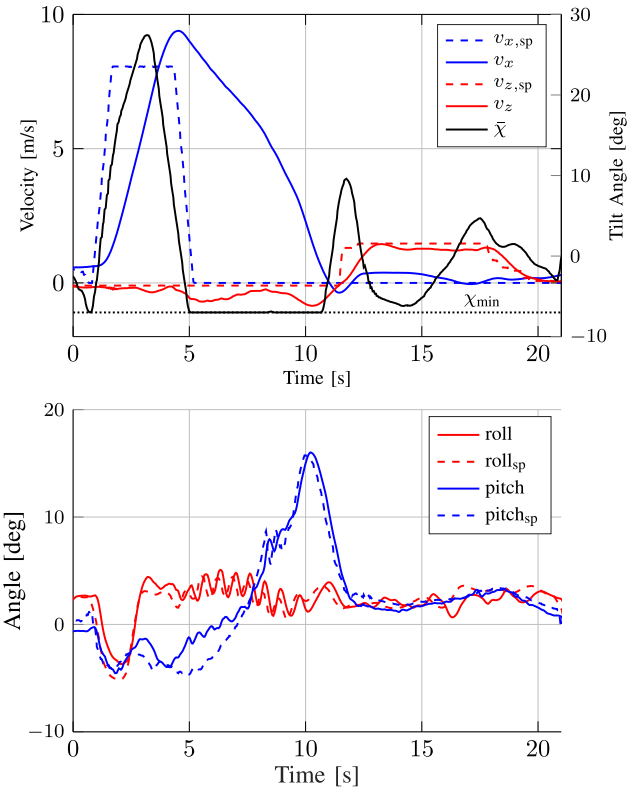

Fig. 10. Plots show measurement data from an outdoor flight with the remote-controlled VTOL aircraft. The lower tilt angle limit $\chi_{\min}$ is also shown to highlight the constraint-awareness of the MPC architecture. First a step in the x_B and then in the z_B direction (body frame) is commanded. The vehicle is able to perform such maneuvers.

When the vehicle is tasked to accelerate, it tilts the propellers forward and maintains zero vertical speed while gaining forward velocity. This behavior is identical to what could be observed in the MPC/FPID comparison in simulation (see Fig. 6). The vehicle also overshoots the given reference in the outdoor flight. Because this overshoot could already be observed in the simulation, it is most likely not caused by a wind gust. A possible explanation is an overestimate of the drag in the dynamics model of the MPC when compared to the actual aircraft. Fig. 10 also shows the roll and pitch of the vehicle during the flight. The pitch reference is tracked very well throughout most phases of the flight. Especially during the deceleration phase pitch angles around 15° are commanded and tracked with little error by the inner-loop attitude controller. This result nicely shows that the developed control allocation works very well as it is able to implement the desired body torques even in challenging flight conditions. The roll angle tracking is less accurate and shows oscillatory behavior. Note that no oscillations were observed in any flights in the verified simulation environment. Also, the real-world roll and pitch tracking tests plotted in Fig. 8 show few signs of oscillations. Given that the roll angle is off by 2° during hover, one possible explanation for the observed behavior is that the PixHawk autopilot was not properly attached to the airframe and, thus, became slightly loose. Due to the geometry of the mounts inside the aircraft, this would lead to some play in the roll axis.

During the deceleration phase the propellers are fully tilted backwards and the vehicle is able to stop after 5 s while maintaining zero vertical speed to within 1 m/s. As soon as the vehicle stops, it tilts the propellers forward to compensate the pitch of the aircraft. The second part of the flight shows the vehicle's ability to track a vertical velocity setpoint. It descends with 1.6 m/s and tracks that reference to within 0.3 m/s. The deviations in the horizontal velocity v_x are most likely due to the mild wind.

Overall, the experiments on the real VTOL aircraft successfully demonstrate the MPC controller's capability to stabilize the aircraft and track a given velocity reference. The flights in simulation further show that the controller's performance extends to flights at higher speeds and that it consistently outperforms the FPID controller. For this reason, the results—although not yet perfect—are very relevant to the community.

VII. CONCLUSION

This article presented the novel application of a MPC controller for VTOL aircraft and demonstrated the performance of the proposed controller in various experiments. The controller is able to fully utilize the vehicle's potential: it is able to operate in both "multicopter mode" or "fixed-wing mode" and it takes advantage of the full range of possible configurations. The model predictive controller is running on a companion computer aboard the vehicle, which communicates with the autopilot. This architecture makes the setup extremely potent since no ground station is needed to run the MPC, and the Pixhawk autopilot runs the attitude controller and custom control allocation at a very high rate. Additionally, a backup controller to cope with MPC failure is present on the autopilot. Experiments in a validated simulation environment and experiments with a real remote-controlled tilt-rotor VTOL aircraft in various scenarios demonstrate the MPC's capability to use the tilting propellers effectively and to outperform the FPID controller. Our work shows that model predictive control is very applicable to tilt-rotor VTOL aircraft. Although challenging,

the changing and nonlinear flight characteristics make a this type of vehicle ideal prospects for MPC. To the best of the authors' knowledge, the presented experimental verification is a first for tilt-rotor fixed-wing VTOL MPC control.

REFERENCES

[1] L. Spannagl and G. Ducard
Control allocation for an unmanned hybrid aerial vehicle
In *Proc. 28th Mediterranean Conf. Control Automat.* Saint-Raphaël, France, 2020, pp. 709–714.

[2] H. Liu, F. Peng, F. L. Lewis, and Y. Wan
Robust tracking control for tail-sitters in flight mode transitions
IEEE Trans. Aerosp. Electron. Syst., vol. 55, no. 4, pp. 2023–2035, Aug. 2019.

[3] B. Li, J. Sun, W. Zhou, C.-Y Wen. , K. H. Low, and C.-K Chen
Transition optimization for a VTOL tail-sitter UAV
IEEE/ASME Trans. Mechatronics, vol. 25, no. 5, pp. 2534–2545, Oct. 2020.

[4] S. A. Emami and A. Rezaeizadeh
Adaptive model predictive control-based attitude and trajectory tracking of a VTOL aircraft
IET Control Theory Appl., vol. 12, no. 15, pp. 2031–2042, 2018.

[5] R. Ritz and R. D'Andrea
, *A Global Strategy for Tailsitter Hover Control*. Cham, Switzerland: Springer, 2018 pp. 21–37. [Online]. Available: https://doi.org/10.1007/978-3-319-51532-8_2

[6] S. Verling, T. Stastny, G. Bättig, K. Alexis, and R. Siegwart
Model-based transition optimization for a vtol tailsitter
In *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3939–3944.

[7] A. T. Tran, N. Sakamoto, M. Sato, and K. Muraoka
Control augmentation system design for quad-tilt-wing unmanned aerial vehicle via robust output regulation method
IEEE Trans. Aerosp. Electron. Syst., vol. 53, no. 1, pp. 357–369, Feb. 2017.

[8] K. Benkhoud and S. Bouallegue
Model predictive control design for a convertible quad tilt-wing UAV
In *Proc. 4th Int. Conf. Control Eng. Inf. Technol*., 2016, pp. 1–6.

[9] R. Donadel, G. V. Raffo, and L. B. Becker
Modeling and control of a tiltrotor UAV for path tracking
In *Proc. 19th World Congr.*, 2014, pp. 3840–3844.

[10] C. Papachristos, K. Alexis, G. Nikolakopoulos, and A. Tzes
Model predictive attitude control of an unmanned tilt-rotor aircraft
In *Proc. IEEE Int. Symp. Ind. Electron*., 2011, pp. 922–927.

[11] M. Ryll, H. H. Bulthoff, and P. R. Giordano
A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation
IEEE Trans. Control Syst. Technol., vol. 23, no. 2, pp. 540–556, Mar. 2015.

[12] M. Kamel *et al.*
The voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle
IEEE Robot. Automat. Mag., vol. 25, no. 4, pp. 34–44, Dec. 2018.

[13] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi
Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers
In *Proc. IEEE Int. Conf. Robot. Automat*., vol. 93, no. 1, 2015, pp. 4006–4013.

[14] M. Ryll, D. Bicego, and A. Franchi
Modeling and control of fast-hex: A fully-actuated by synchronized-tilting hexarotor
In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., 2016, pp. 1689–1694.

[15] X. Fang, Q. Lin, Y. Wang, and L. Zheng
Control strategy design for the transitional mode of tiltrotor UAV
In *Proc. 10th IEEE Int. Conf. Ind. Informat.*, 2012, pp. 248–253.

[16] F. Cakici and M. K. Leblebicioglu
Control system design of a vertical take-off and landing fixed wing UAV
IFAC-PapersOnLine, vol. 49, no. 3, pp. 267–272, 2016.

[17] L. R. Nardizzi, M. Y. Tarng, and R. J. Parker
Optimal and suboptimal control synthesis for minimum time VTOL transition
IEEE Trans. Aerosp. Electron. Syst., vol. AES- 7, no. 3, pp. 506–520, May 1971.

[18] R. Mehra, R. Prasanth, and S. Gopalaswamy
Xv-15 tiltrotor flight control system design using model predictive control
In *Proc. IEEE Aerosp. Conf.*, Aspen, CO, USA, Mar. 21–28, 1998, vol. 2, pp. 139–148.

[19] O. Tekinalp, T. Unlu, and I. Yavrucuk
Simulation and flight control of a tilt duct UAV
In *Proc. Model. Simul. Technol. Conf.*, 2009.

[20] Z. Liu, S. Tang, M. Li, and J. Guo
Optimal control of thrust-vectored VTOL UAV in high-manoeuvering transition flight
Aeronautical J., vol. 122, no. 1250, pp. 598–619, 2018.

[21] M. Allenspach and G. J. J. Ducard
Model predictive control of a convertible tiltrotor unmanned aerial vehicle
In *Proc. 28th Mediterranean Conf. Control Autom.*, 2020, pp. 715–720.

[22] C. Papachristos, K. Alexis, and A. Tzes
Model predictive hovering-translation control of an unmanned tri-tiltrotor
In *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5425–5432.

[23] G. Ducard and M.-D. Hua
Modeling of an unmanned hybrid aerial vehicle
In *Proc. IEEE Conf. Syst. Control*, 2014, pp. 1011–1016.

[24] L. Bauersfeld and G. Ducard
Fused-PID control for tilt-rotor VTOL aircraft
In *Proc. 28th Mediterranean Conf. Control Autom. Saint-Raphaël*, France, 2020, pp. 703–708.

[25] K. Rudin, G. J. J. Ducard, and R. Y. Siegwart
Active fault tolerant control with imperfect fault detection information: Applications to UAVs
IEEE Trans. Aerosp. Electron. Syst., vol. 56, no. 4, pp. 2792–2805, Aug. 2020.

[26] "ℓ_1-norm methods for convex-cardinality problems
Univ. Stanford, Stanford, CA, USA. [Online]. Available: https://stanford.edu/class/ee364b/lectures/l1_slides.pdf

[27] M. Okamoto, K. Yasuda, and A. Azuma
Aerodynamic characteristics of the wings and body of a dragonfly
J. Exp. Biol., vol. 199, no. Pt 2, pp. 281–294, 1996.

[28] B.-C Wang
Immersed body flow
Univ. Manitoba: Fluid Mechanics Appl.
Ch. 4, 2017. [Online]. Available: https://home.cc.umanitoba.ca/~wang44/Courses/MECH3492/Handout_Ch4.pdf

[29] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari
FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs
Int. J. Control, vol. 93, no. 1, pp. 13–29, 2020.

[30] A. Domahidi and J. Jerez
Forces professional
Embotech AG, 2014–2019. [Online]. Available: https://embotech.com/FORCES-Pro

Leonard Bauersfeld received the B.S. degree in mechanical engineering from ETH Zürich, Zürich, Switzerland, in 2018. He is currently working toward the master's degree in robotics, systems and control from Robotics and Perception Group, University of Zurich.

Within the context of this article, he was affiliated with the Institute of Dynamic Systems and Control, ETH Zürich. His current research interests include aerodynamic modeling and flight control for UAVs.

Lukas Spannagl received the B.Sc. degree in mechanical engineering, in 2018 from ETH Zürich, Zürich, Switzerland, where he is currently working toward the M.Sc. degree in robotics, systems and control with Institute for Dynamic Systems and Control.

He is currently affiliated with the Institute for Dynamic Systems and Control . His current research interests include optimal guidance and control as well as thrust vectored vehicles.

Guillaume J. J. Ducard received the master's degree in electrical engineering and the Doctoral degree focusing on flight control for unmanned aerial vehicles (UAVs) from ETH Zürich, Zürich, Switzerland, in 2004 and 2007, respectively.

He completed his two-year Postdoctoral course in 2009 from ETH Zürich, focused on flight control for UAVs. He is currently an Associate Professor with the University Côte d'Azur, France, and guest scientist with ETH Zürich. His research interests include nonlinear control, estimation, and guidance applied to UAVs.

Christopher H. Onder received the Diploma in mechanical engineering and Doctoral degree in Doctor of technical sciences from ETH Zürich, Zürich, Switzerland.

He is a Professor with the Institute for Dynamic Systems and Control, Department of Mechanical Engineering and Process Control, ETH Zürich. He heads the Engine Systems Laboratory. He has authored and coauthored numerous articles and a book on modeling and control of engine systems. His research interests include engine systems modeling, control and optimization with an emphasis on experimental validation, and industrial cooperation.

Dr. Onder was the recipient of the BMW scientific award, the ETH medal, the Vincent Bendix award, the Crompton Lanchester Medal, and the Arch T. Colwell award. Additionally, he received several times the Watt d'Or, the energy efficiency price of the department of energy of Switzerland, for his projects.