2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Inversion Based Direct Position Control and Trajectory Following for Micro Aerial Vehicles

Markus W. Achtelik, Simon Lynen, Margarita Chli and Roland Siegwart

***Abstract*— In this work, we present a powerful, albeit simple position control approach for Micro Aerial Vehicles (MAVs) targeting specifically multicopter systems. Exploiting the differential flatness of four of the six outputs of multicopters, namely position and yaw, we show that the remaining outputs of pitch and roll need not be controlled states, but rather just need to be known. Instead of the common approach of having multiple cascaded control loops (position - velocity - acceleration/attitude - angular rates), the proposed method employs an outer control loop based on dynamic inversion, which directly commands angular rates and thrust. The inner control loop then reduces to a simple proportional controller on the angular rates. As a result, not only does this combination allow for higher bandwidth compared to common control approaches, but also eliminates many mathematical operations (only one trigonometric function is called), speeding up the necessary processing especially on embedded systems. This approach assumes a reliable state estimation framework, which we are able to provide with through previous work. As a result, with this work, we provide the missing elements necessary for a complete approach on autonomous navigation of MAVs.**

I. Introduction

Inspired by the work of Wang et. al [1], this paper describes a two-loop controller design for rotor-based MAVs which is based on nonlinear dynamic inversion. Recently, Mellinger and Kumar showed in [2] that a quadcopter is only differentially flat on four outputs, namely position and yaw. This means that the four inputs of thrust and angular acceleration can be expressed solely using the flat outputs and their derivatives. The remaining outputs of roll and pitch, do not explicitly appear and are just functions of the desired accelerations. Therefore, there is no reason for these two states to be *controlled* states, but rather, they just need to be *known* – which we can perfectly achieve with our state estimation framework [3], [4] presented in earlier work.

Relevant to this paper is the work on controlling quadcopters for performing precision maneuvers, such as flips and balancing an inverted pendulum while flying [5], [6], [7], [8]. These works have been carried out at the Flying Machine Arena (FMA) of ETH Zurich equipped with a high-precision external tracking system. The difference of these approaches and the one by Mellinger and Kumar is that the FMA works rely on "perfect" state estimation from a motion capture system, and thus, do not focus on how to exploit information from inertial sensors onboard the MAV.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7) under grant agreement n.266470 (myCopter). Markus Achtelik and Simon Lynen are currently PhD students, Margarita Chli is deputy director and senior researcher at the Autonomous Systems Lab (ASL), and Roland Siegwart is full professor at the ETH Zurich and head of the ASL. (email: {markus.achtelik, simon.lynen, margarita.chli}@mavt.ethz.ch, r.siegwart@ieee.org).

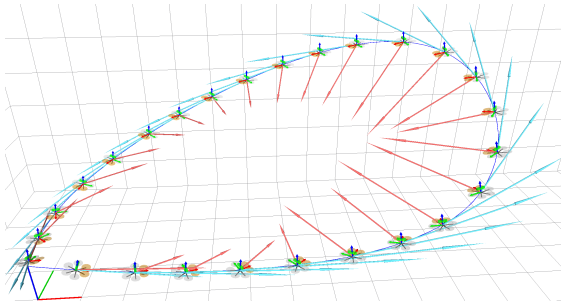

Fig. 1: Trajectory based on 11^{th} order polynomials . The local coordinate frames along the trajectory show the vehicle's attitude, while the blue and red arrows show the velocity and acceleration respectively.

In this paper, we show how to directly command angular velocities in an outer control loop and a simple angular rate controller as the inner control loop. The motivation for using this two-loop design is manifold. Firstly, by skipping the common cascaded position-velocity-attitude loop, we obtain increased bandwidth, allowing for aggressive maneuvers. Secondly, as shown in the subsequent sections, the mathematical formulation of this approach gets substantially simplified and thus is computationally very efficient: throughout the whole control loop, there is only one $atan2()$ operation that has to be computed, which makes the approach perfectly suited for computationally limited micro-processors. Thirdly, this approach allows for a separation of the position control problem from vehicle-specific parameters, which promotes general applicability: a rate loop can be easily implemented on any multicopter-like platform with a simple controller based on gyro readings (i.e. no sophisticated state estimation required for the inner loop). Thrust is simply expressed in terms of acceleration since the mass of the vehicle is known. As a result, this approach hides vehicle-specific parameters from the position control loop, permitting transparent employability on different MAVs, be it a quadcopter, or a hexacopter with/without motor failure.

Drawing inspiration by [1] and [2] the contributions of this work are the following. We extend the two-loop design proposed in [1] and adapt it to our vehicle with the given control inputs. We show how to implement the low level rate control loop for a hexacopter and how we can estimate vehicle-specific parameters, such as the moment of inertia and propeller constants. Then, we show our extension to the minimum snap trajectories presented by Mellinger and Kumar [2]. Lastly, we show how using the proposed controller, we can exploit the states computed by our powerful state estimation framework [3], [4].

The remainder of the paper is organized as follows: after a brief description of our helicopter platform, we present the key properties of our state estimation framework and how the overall system is implemented. Section III describes our controller in detail and Section IV how suitable trajectories are generated. Results are shown in Section V and conclusions are drawn in Section VI.

II. System Setup

Our setup consists of a micro helicopter equipped with an Inertial Measurement Unit (IMU), a monocular down-looking camera and an onboard computer. The helicopter is the "FireFly" hexacopter from Ascending Technologies (Fig. 4). Compared to the "AscTec Pelican" it has improved vibration damping for the sensors (camera, Flight Control Unit (FCU)) and can tolerate failure of one rotor. The FCU features two 60 MHz ARM 7 microprocessors, the Low-Level Processor (LLP) and the High-Level Processor (HLP), as well as an Inertial Measurement Unit (IMU). The LLP is delivered as a black box, and responsible for all low level tasks that are required for manual flight operation, while the HLP is dedicated to custom user code. It can access all relevant data (angular rates, linear acceleration, rotor speeds) from the LLP over a high-speed interface almost delay-free. In return, it can access the LLP on different control levels: On the highest level, it can send GPS waypoints. On an intermediate level it can send attitude, yaw-rate and thrust commands. In the following, we work with the lowest command level: we send individual rotor speed commands to the LLP, which forwards those directly to the motor controllers.

For the computationally more demanding tasks like our Extended Kalman Filter for state estimation and our visual navigation framework [4], [3], the FireFly is equipped with a "MasterMind" embedded 1.86 GHz Core2Duo onboard computer from Ascending Technologies.

A. State Estimation

For our control approach it is key to have a fast and reliable framework for state estimation. For this purpose, we use the powerful EKF (Extended Kalman Filter) based state estimation framework described in [4], [3]. For completeness, here we briefly summarize the essentials and point out the key properties needed for the controller.

Position or pose measurements are fused with IMU readings (linear acceleration and angular velocity) in an EKF framework. The findings of [9], [10], [11] are applied to not only estimate the pose and velocity of the MAV, but also the sensor biases, the scale of the position estimates and the (inter-sensor) calibration between the IMU and the pose/position sensor *in real-time*.

B. MAV Platform and Onboard Hardware

The state variables of the filter may vary according to the application and sensor setup. However, we identified a part of the state that stays constant, which is of interest for this work. It is composed of the position of the IMU $\boldsymbol{p}$ in the world frame, its velocity $\boldsymbol{v}$ and its attitude quaternion q describing the rotation of the IMU w.r.t. the world frame. In the following, we use the rotation matrix $\boldsymbol{R}(q)$, or short $\boldsymbol{R}$, denoting the attitude of the helicopter. The columns of $\boldsymbol{R}$ can also be interpreted as the unit vectors of the helicopter's body coordinate system: $\boldsymbol{R} = \begin{bmatrix}\boldsymbol{x}_B & \boldsymbol{y}_B & \boldsymbol{z}_B\end{bmatrix}$. Fig. 2 shows the coordinate frames with the respective variables. We also add the gyro and acceleration biases $\boldsymbol{b}_\omega$ and $\boldsymbol{b}_a$ to the state, which yields the following part of the filter state vector $\boldsymbol{x}_f$:

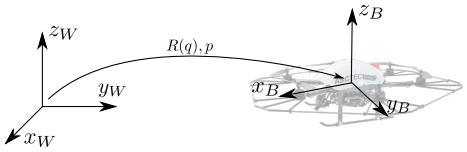

Fig. 2: Setup depicting the helicopter body w.r.t. a world reference frame. $\boldsymbol{R}$ or q denote the attitude of the helicopter and $\boldsymbol{p}$ denotes the position w.r.t. the world frame

$$\boldsymbol{x}_f = \begin{bmatrix}\boldsymbol{p}^T & \boldsymbol{v}^T & q & \boldsymbol{b}_\omega^T & \boldsymbol{b}_a^T & \dots\end{bmatrix}^T \tag{1}$$

For a detailed description and analysis of the EKF framework, we refer to [4], [3].

C. System Overview

How the whole system works and how its components interact, can be seen in Fig. 3. Since IMU readings are available at 1 kHz at the HLP of the FCU, we compute the less expensive EKF state prediction on the HLP at the same rate. This provides estimates of $\boldsymbol{p}$, $\boldsymbol{v}$ and q to the position controller and lets us correct the IMU readings, also needed by the controller, for their biases. The controller computes individual rotor speed commands and sends those to the LLP. The covariance propagation and update, as well as the EKF measurement update stage run on the Core2Duo computer due to their complexity. This approach lets us handle the fast dynamics of the helicopter, while allowing the computation of an ideal Kalman gain based on the uncertainties of the state and the measurements. More details on this distribution of EKF tasks can be found in [3]. Smooth trajectories (Section IV) are generated on the Core2Duo computer and are used as reference and feed forward signal for the controller.

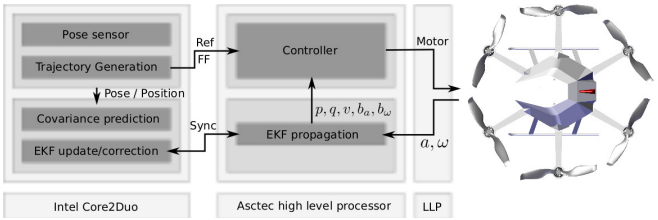

Fig. 3: System overview: the computational demanding tasks are executed on the Core2 computer, while the less intensive and time critical parts are executed on the high-level micro processor at a rate of 1 kHz. The EKF propagation part provides a current state estimate and IMU bias corrections to the controller, which obtains its reference (Ref) and feed forward (FF) signals from the trajectory generator

III. Controller

Our proposed two-loop controller design is comprised of an outer (position) control loop and an inner (rate)

loop. Following an explanation of the approach, we show how control allocation works with the hexacopter and how vehicle-specific parameters can be estimated.

A. Direct Rate Control

In this section, we derive a position controller, that directly commands angular rates, eliminating the need for the attitude control loop commonly used.

1) Dynamic Inversion: With the method of dynamic inversion, feedback linearization is applied to our desired outputs (position $\boldsymbol{p}$, yaw ψ), transforming the nonlinear system to a linear one. This lets us apply well known and understood linear control system techniques [12], [1], and it furthermore facilitates trajectory generation. We define *pseudo* control commands $\boldsymbol{u}$ for the linear system, which we finally turn into control commands available from the vehicle. We ignore air drag here and will explain in Section III-B how this can be handled.

The velocity $\boldsymbol{v}$ is simply the first derivative of the position $\boldsymbol{p}$. The acceleration $\boldsymbol{a}$ (seen in the world frame) depends on the thrust T, attitude $\boldsymbol{R}$ and gravity g:

$$\boldsymbol{a} = \boldsymbol{R} \cdot \begin{bmatrix} 0 & 0 & T \end{bmatrix}^T - \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T \tag{2}$$

Taking the first derivative of $\boldsymbol{a}$, i.e. the jerk $\boldsymbol{j}$, and using the chain differentiation rule with $\dot{\boldsymbol{R}} = \boldsymbol{R} \cdot \lfloor \boldsymbol{\omega} \times \rfloor$, where $\boldsymbol{\omega}$ expresses body-fixed angular velocities, yields:

$$\boldsymbol{j} = \boldsymbol{R} \cdot \lfloor \boldsymbol{\omega} \times \rfloor \cdot \begin{bmatrix} 0 & 0 & T \end{bmatrix}^T + \boldsymbol{R} \cdot \begin{bmatrix} 0 & 0 & \dot{T} \end{bmatrix}^T \tag{3}$$

Due to the cross product ($\lfloor \times \rfloor$), only the x- and y- components of $\boldsymbol{\omega}$ remain, so that we can write:

$$\begin{bmatrix} \omega_y & -\omega_x & \dot{T} \end{bmatrix}^T = (\boldsymbol{R}^T \cdot \boldsymbol{j})/T \tag{4}$$

With $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{x}_B & \boldsymbol{y}_B & \boldsymbol{z}_B \end{bmatrix}$, i.e. the unit vectors defining the body coordinate system of our helicopter, we obtain the desired inversion to vehicle controls

$$\omega_x = -\boldsymbol{y}_B^T \cdot \boldsymbol{j}/T \tag{5}$$

$$\omega_y = \boldsymbol{x}_B^T \cdot \boldsymbol{j}/T \tag{6}$$

$$\dot{T} = \boldsymbol{z_B}^T \cdot \boldsymbol{j} \tag{7}$$

ω_z is simply, as derived in [2]:

$$\omega_z = \dot{\psi} \cdot \boldsymbol{z}_W^T \cdot \boldsymbol{z}_B \tag{8}$$

B. Outer Control Loop

With Eq. (5) to Eq. (8) at hand, we can compute feed forward control signals to follow a specified trajectory generated with the methods in Section IV. However, as there are model uncertainties and disturbances, we need an error controller to keep the vehicle on the desired trajectory.

For ψ, this can be achieved with a simple P(I)- controller generating $\dot{\psi}$ and does not need any further investigation. From our state estimation framework (Section II-A), we get estimates for position $\boldsymbol{p}$, velocity $\boldsymbol{v}$. Accelerations $\boldsymbol{a}_B$ expressed in body coordinates can be measured with the accelerometers.These can be bias corrected and expressed w.r.t. world coordinates with our estimates of the acceleration bias $\boldsymbol{b}_a$ and attitude $\boldsymbol{R}$. That way, we can design a state feedback controller for every single axis, that turns errors of $\boldsymbol{p}$, $\boldsymbol{v}$ and $\boldsymbol{a}$ w.r.t. the desired reference into *pseudo* jerk ($\boldsymbol{j}$) commands. Using the inversion from Eq. (7) we can finally compute the desired vehicle commands.

Having the *measured*, rather than the not (model based) estimated acceleration in the state feedback loop has the nice property that disturbances like wind gusts or air drag can be directly measured and be compensated for. In contrast, with common cascaded approaches, disturbances would first integrate to velocity before the controller can react; and would be converted back into an appropriate control command.

The inversion from Eq. (7) works fine for the angular rates $\boldsymbol{\omega}$, but this inversion also computes a desired change of thrust, i.e. a change of rotor speed. However, this entails some problems, since thrust is a direct function of the rotor speeds and the latter is the only input we have. A naive solution would be to integrate $\dot{T}$ over the control period and obtain a thrust command from that. A more well-founded solution can be best explained via the following example since it is a little counter-intuitive: consider the helicopter hovering and leveled. We have the thrust vector with thrust T pointing downwards with the system expected to have 2nd order integration behaviour. The situation changes when the helicopter starts turning around its x- and/or y- axis: as evident from Eq. (3), angular velocity ω_x, ω_y causes a *change* of acceleration, resulting in 3rd order behaviour. However, the thrust T (2nd order behaviour) remains at that instant. Looking back at the pseudo commands that we apply to the system, we come to the conclusion that we have two different system inputs: simplified around the hovering point, there is acceleration caused by the thrust acting in the z_W axis and jerk caused by ω_x, ω_y acting in the x_W and y_W axis. As soon as the helicopter is not leveled anymore, both inputs get combined.

We therefore define the two 3-dimensional pseudo controls $\boldsymbol{u}_a$ for acceleration and $\boldsymbol{u}_j$ for jerk, both expressed in world coordinates. For the pseudo-commands, the axes are decoupled so that we can have dedicated controllers for each axis. As a result, the linear 3rd order system now is:

$$\dot{\boldsymbol{x}} = \boldsymbol{A} \cdot \boldsymbol{x} + \boldsymbol{B} \cdot \boldsymbol{u} \quad , \tag{9}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} p \\ v \\ a \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} u_a \\ u_j \end{bmatrix} \tag{10}$$

The decoupling of the inputs works that way, since we see from Eq. (4) that $\boldsymbol{\omega}$ relating to u_j is independent from the change of thrust. This system can now be stabilized with a linear state feedback error controller $\boldsymbol{K}$ that can be designed by pole placement or LQR techniques. We obtain for the pseudo commands:

$$\boldsymbol{u}_{\text{err}} = -K \cdot (\boldsymbol{x}_{\text{ref}} - \boldsymbol{x}) \tag{11}$$

Where $\boldsymbol{x}_{\text{ref}}$ is the reference trajectory, which is either $\mathbf{0}$ for hovering or a trajectory that we generate with the method in

Section IV. With $\boldsymbol{u}_{\text{ff}}$ being the feed forward signal from the trajectory generation, we obtain our final control signal:

$$\boldsymbol{u} = \boldsymbol{u}_{\text{err}} + \boldsymbol{u}_{\text{ff}} \ . \tag{12}$$

Rearranging the components of $\boldsymbol{u}_x$, $\boldsymbol{u}_y$ and $\boldsymbol{u}_z$ into our 3D pseudo commands $\boldsymbol{u}_a$ and $\boldsymbol{u}_j$, we can compute all vehicle controls according to Eq. (2), (5), (6) and (8) that we can work with in the next section:

$$\omega_z = \dot{\psi} \cdot \boldsymbol{z}_W^T \cdot \boldsymbol{z}_B \qquad T = \boldsymbol{z}_B^T \cdot \boldsymbol{u}_a + g \tag{13}$$

$$\omega_x = -\boldsymbol{y}_B \cdot \boldsymbol{u}_j / T \qquad \omega_y = \boldsymbol{x}_B \cdot \boldsymbol{u}_j / T \tag{14}$$

C. Inner Loop and Control Allocation

For the inner angular rate control loop on $\boldsymbol{\omega}$, we use a simple proportional controller for each axis, that command angular accelerations $\dot{\boldsymbol{\omega}}$. In order to convert this into rotor speed commands, we need to have a look at the inner dynamics of the helicopter, which we will show for our case of a hexacopter. This would also be the place to implement the reactions towards rotor failure.

Each rotor produces a force $F = k_n \cdot n^2$, where n is the rotational speed of a rotor and k_n a rotor constant. Also, depending on F, each rotor creates a torque M around its z-axis: $M = k_m \cdot F$, where k_m is again a rotor constant [1]. We assume that the moment of inertia matrix $\boldsymbol{I}$ of the helicopter has only diagonal entries. The mass of the helicopter is denoted as m. Looking at the geometry of the hexacopter (Fig. 4), we can now express the forces and torques as follows, with l being the boom length, $s = \sin(30°)$ and $c = \cos(30°)$:

$$\underbrace{\begin{bmatrix}\dot{\boldsymbol{\omega}}\\ T\end{bmatrix}}_{\boldsymbol{u}_i} = \mathcal{I}^{-1} \cdot \boldsymbol{K} \cdot \overbrace{\underbrace{\begin{bmatrix} s & 1 & s & -s & -1 & -s \\ -c & 0 & c & c & 0 & -c \\ -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}}_{\boldsymbol{A}} \cdot \underbrace{\begin{bmatrix} n_1^2 \\ \vdots \\ n_6^2 \end{bmatrix}}_{\boldsymbol{n}}}^{\text{forces, thrust}} \tag{15}$$

$$\mathcal{I} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & m \end{bmatrix}; \ \boldsymbol{K} = \text{diag}(\begin{bmatrix} l\,k_n & l\,k_n & k_n\,k_m & k_n \end{bmatrix}) \tag{16}$$

$\boldsymbol{K}$ determines a parameter matrix, where we already factored out all vehicle specific parameters, whereas $\boldsymbol{A}$ determines an allocation matrix for a generic hexacopter as shown in Fig. 4.

To obtain the desired rotor speeds n_i, we need to solve for $\boldsymbol{n}$ and compute its element-wise square root. This is straightforward for a regular quadcopter, as $\boldsymbol{A}$ is directly invertible, but is not the case for the hexacopter. As a solution, we propose to compute the pseudo-inverse $\boldsymbol{A}^\dagger$. This minimizes the Euclidean norm of $\boldsymbol{n}$ and is thus the most energy efficient solution. Since we factored out the vehicle-specific parameters beforehand, $\boldsymbol{A}^\dagger$ only needs to be computed once and can be stored. We finally obtain for $\boldsymbol{n}$:

$$\boldsymbol{n} = \boldsymbol{A}^\dagger \cdot \boldsymbol{K}^{-1} \cdot \mathcal{I} \cdot \boldsymbol{u}_i \tag{17}$$

These operations are fast to compute, since $\boldsymbol{K}^{-1}$ and $\mathcal{I}$ are diagonal matrices and every entry just affects a single row of $\boldsymbol{A}^\dagger$. Given $\boldsymbol{I}$, m and $\boldsymbol{K}$, we can now choose the gains of the rate controllers in a "metric space". How to obtain $\boldsymbol{K}$ and $\mathcal{I}$ is shown in the next section.

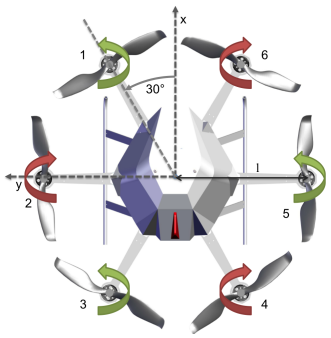

Fig. 4: Our FireFly hexacopter with body coordinate system (x forward, y left, z up), rotor turning directions and rotor geometry.

D. System Parameter Estimation

While only a few parameters, namely l and m, can be easily and accurately measured, this is complicated and prone to errors for the moment of inertia matrix $\boldsymbol{I}$ and the rotor constants k_m and k_n. In addition, when measured separately, errors might add up and yield incorrect values for the mapping between $\boldsymbol{n}$ and $\boldsymbol{u}_i$. Since we already factored out all vehicle specific parameters in Eq. (15) and the matrices $\boldsymbol{K}$, $\mathcal{I}$ are diagonal, there is only one parameter per row left that maps $\boldsymbol{n}$ to $\boldsymbol{u}_i$:

$$\boldsymbol{u}_i = \text{diag}\,(\boldsymbol{k}) \cdot \boldsymbol{A} \cdot \boldsymbol{n}; \quad \boldsymbol{k} = \begin{bmatrix} k_{\omega_x} & k_{\omega_y} & k_{\omega_z} & k_T \end{bmatrix} \tag{18}$$

We can estimate the parameters in question by a quick manually-controlled flight exhibiting some excitation to the vehicle. All that is needed are acceleration measurements in the body z-axis, angular rate measurements from the gyros as well as the rotor speeds. If our state estimation framework is run in parallel, we can also compensate for the gyro and acceleration biases. Therefore, we assume for the following that these measurements are already bias compensated. Parameters like k_n appear for many axes, but for simplicity, we estimate the mappings separately for each axis. In the following, $\boldsymbol{A}_i$ denotes the i-th row of the allocation matrix $\boldsymbol{A}$.

For k_T, there is just a linear mapping between $F_T = \boldsymbol{A}_4 \cdot \boldsymbol{n}$ and T. Simple averaging of measurements first does not handle noise well, and second, was shown to yield wrong results [13]. We therefore use the maximum likelihood estimator proposed in [13] to iteratively find the correct value for k_T. Noise parameters for a_z can be found in the data sheet. The rotor speed measurements itself would be accurate, but we have to account for quantization noise since we can unfortunately obtain n_i only in a range from $1 \ldots 200$. The mapping c_{rpm} between these measurements and revolutions per minute (rpm) is $c_{rpm} = 40$ for both the AscTec Hummingbird and Pelican helicopters, and $c_{rpm} = 50.75$ for the FireFly (constants provided by AscTec). Converted to rad/s we obtain the motor constant c_m and noise σ_m:

$$\sigma_m = \frac{1}{\sqrt{12}} \cdot c_m; \ c_m = c_{rpm}/60 \cdot 2\pi \tag{19}$$

We cannot determine $k_{\omega_{x,y,z}}$ directly as above, since we cannot measure $\dot{\boldsymbol{\omega}}$. However, we can filter it with a two dimensional Extended Kalman Filter. The problem is nonlinear since the noise of the system input is the squared rotor speed. We show our method exemplary for the x-axis, but it is applicable in the same manner for the remaining axes. We define our state as $x = \begin{bmatrix}\omega_x & k_{\omega_x}\end{bmatrix}^T$, and the following differential equations govern the state (for simplicity, we omit the subscripts):

$$\dot{\omega} = k \cdot \boldsymbol{A}_1 \cdot \boldsymbol{n}; \quad \boldsymbol{n} = \left[(n_1 + n_n)^2 \dots (n_6 + n_n)^2\right]^T \quad (20)$$

$$\dot{k} = 0 \text{ (does not change over time)} \quad (21)$$

Where n_i, $i = 1 \dots 6$ are the rotor speeds and n_n is zero mean white Gaussian noise as obtained in Eq. (19). For the measurement, we have $z = \omega$. With these equations at hand, we can perform the standard EKF procedure as described in [9] in order to find a good estimate for k.

In practice, only a few seconds of flight with varying $\boldsymbol{\omega}$ and a_z were necessary for finding good values for $\boldsymbol{k}$. An example for the estimation of k_{ω_x} can be seen in Fig. 5. We determined the values of $\boldsymbol{k}$ as follows, which may serve as a starting point, but is subject to vary, depending on the vehicle setup:

$$\boldsymbol{k} = \begin{bmatrix}5.5 \cdot 10^{-5} & 4.5 \cdot 10^{-5} & 2.5 \cdot 10^{-6} & 6.7 \cdot 10^{-6}\end{bmatrix}$$

The different values for the x- and y- axis are no surprise: the mass of the battery is more centered around the x-axis of the helicopter, causing different moment of inertia.

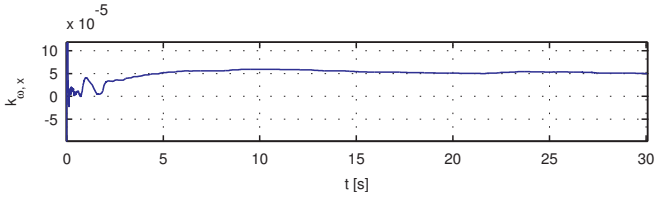

Fig. 5: Estimation of the parameter k_{ω_x}. The helicopter was manually flown with rates of $\approx \pm 2\,\mathrm{rad/s}$

E. Remarks on Actuator Saturation

We do not provide an in depth study of actuator saturation here, as this would go beyond the scope of this paper. However, we can have a look at actuator saturation at two different levels: In the planning or trajectory generation phase, one can already incorporate saturation during the time optimization described in Section IV with some safety margin, assuming moderate disturbances. As a result, segment times can be adjusted accordingly. In case of stronger disturbances, techniques like Pseudo Control Hedging (PCH) as suggested in [1] could be applied. In this case, since our trajectory generator is fast, the trajectory can be re-planned and adapted to the current situation.

IV. Trajectory Generation

For the controller described in the previous section, we need to generate smooth trajectories in a way that the vehicle is able to follow. We decided to adapt the *minimum snap trajectory* approach by Mellinger and Kumar [2] to our needs. Their approach uses a set of N^{th} order polynomials with more degrees of freedom than equality constraints (like start/end point), and leaves the remaining constraints to a solver, optimizing a quadratic cost function and inequality constraints. We want to have the feed forward signals for $\boldsymbol{\omega}$ continuous and differentiable (c.f. Eq. (4)) and thus require the derivative of the jerk or 4^{th} derivative of the position, i.e. the snap, to be continuous.

We extend our approach presented in [14], where we just needed a single polynomial segment, to planning a path with multiple segments, enabling smooth flights through multiple waypoints. These waypoints can either be user-defined or set from a path planning or obstacle avoidance algorithm. In contrast to [2], we chose to optimize over the integral of the squared norm of the acceleration instead of snap, minimizing the energy that the helicopter needs. Compared to snap, acceleration directly translates into permanent additional thrust that all the motors have to provide, while snap just causes particular motors to spin up/down quickly. We kept our implementation flexible in order to be able to experiment with velocity, jerk or snap as well.

With our outer control loop from the previous section working on each axis separately, we can plan trajectories also separated for each axis. It is only important, that the segments for each axis stay synchronized, i.e. have the same start and end times. Instead of an *overall* path time, we work with M *segment* times $T_m;\ m = 1 \dots M$. This simplifies the math and computational effort for the cost function, and avoids numeric problems for large values of t (10^{th} power of t or higher) For optimizing a segment over the o^{th} derivative of the position, we obtain the following quadratic optimization problem, subject to some equality constraints which we explain below:

$$f_m(t) = \boldsymbol{t} \cdot \boldsymbol{c}_m\ ;\ \boldsymbol{t} = \begin{bmatrix}1 & t & t^2 \dots t^{N-1}\end{bmatrix} \quad (22)$$

$$\min \int_0^{T_m} \left\| \frac{d^o f_m(t)}{dt^o} \right\| \quad (23)$$

Where $f(t)$ is a N^{th} order polynomial with $N+1$ coefficients $c_0 \dots c_N$ of the position. Start-, end- or intermediate equality constraints of the o^{th} derivative at time t for a *segment* can be formulated as follows. Usually, at the beginning and at the end of the *path*, this is the position and all its derivatives set to zero, while there is only a position constraint with its derivatives left free for intermediate points.

$$\frac{d^o f_m(t)}{dt^o} = b_{m,o} \quad (24)$$

What is left is to ensure continuity between the segments, where the derivatives of the position (or even the position) were left free, and can be formulated as the following equality constraint:

$$\frac{d^o f_m(T_m)}{dt^o} - \frac{d^o f_{m+1}(0)}{dt^o} = 0 \quad (25)$$

A nice property of the above optimization problem is that there are no inequality constraints, which makes it solvable with a closed form solution. However, this still

depends on a fixed time T_m to travel along the trajectory. Nonlinear optimization with the time as additional parameter and inequality constraints are costly (and numerically problematic), therefore we were seeking for a simple and fast solution. We solve the aforementioned problem with a conservative estimate for each T_m. A good starting point is to assume average maximum speed over the straight connection between the waypoints. We compare the extrema to pre-defined maximum values for each derivative on each segment. Then, we scale the path time according to the derivative that is closest to its maximum value:

$$c(o) = \text{abs}((\frac{d^o f(t)}{dt^o})/c_{o,max});\ n = 1 \dots 4 \tag{26}$$

$$T_{m,new} = \max(c(o)) \cdot T^{(1/\text{argmax}(c(o)))} \tag{27}$$

We then recompute the optimization problem with the new path time $T_{m,new}$, which is fast since we do not have inequality constraints in the optimization problem. It is important to note, that this time-scaling has to be done *synchronized* for the 3 coordinate axes that belong to one segment, such that their segment times are equal.

After having obtained a feasible path, we obtain the reference position $\boldsymbol{p}$, velocity $\boldsymbol{v}$, acceleration $\boldsymbol{a}$ and jerk $\boldsymbol{j}$ from the set of polynomials and their derivatives. We then update the control commands with the feed forward signal $\boldsymbol{u}_{\text{ff}} = \begin{bmatrix}\boldsymbol{a}^T & \boldsymbol{j}^T\end{bmatrix}^T$ according to Eq. (12). With Eq. (13), $\boldsymbol{a}$ gets fed forward to the thrust, and with Eq. (14), $\boldsymbol{j}$ gets fed forward to the angular rates around the x- and y- axis.

V. Experiments

We performed all experiments in the Flying Machine Arena [5], which is equipped with a Vicon motion capture system providing us with sub-millimeter pose accuracy that we use as ground-truth in comparisons. We also use the Vicon system for the pose measurements of our state estimation framework in order to isolate the evaluation on the performance of the proposed controller from potential visual SLAM inaccuracies. For a more realistic test setup, we introduce uncertainty in the Vicon readings used as input in the system, to reflect the typical values when flying our MAV outdoors with vision based navigation relatively close to the ground: we reduced the Vicon rate to 20 Hz and added noise with $\sigma = 1\,\text{cm}$.

A. Hovering and Simple Waypoint Following

For the first experiment of desired performance corresponding to hovering at the same spot, we obtained a RMS error in position of 2.2 cm with a maximum of 6.1 cm. In a subsequent experiment, we used our trajectory generator explained in Section IV to fly over a distance of 3 m. The path is slightly diagonal to demonstrate the successful time synchronization between the decoupled axes. The result can be seen in Fig. 6, where we have small errors in position and the vehicle follows the desired trajectory closely. Note that despite the vehicle made large changes in attitude, its position in the z-axis stays constant, showing that our allocation and feed-forward commands work well.

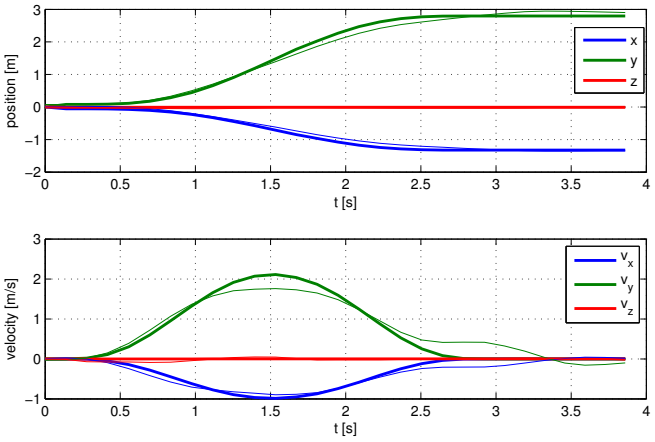

Fig. 6: Diagonal flight over a distance of 3 m with a speed of up to $2\,\text{m/s}$ and acceleration up to $2.5\,\text{m/s}^2$. The thicker lines illustrate the desired trajectory, while the thin lines show the actual position (top) / velocity (bottom). The position in the z- axis remains constant despite large changes in the attitude.

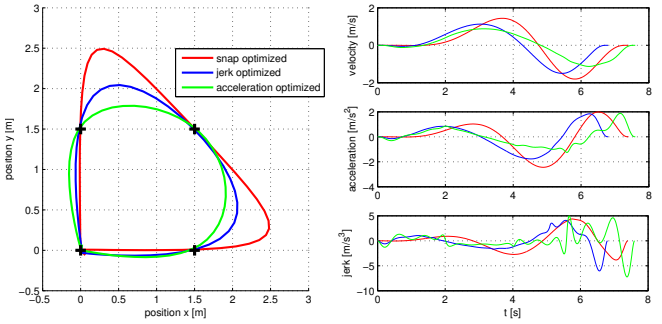

Fig. 7: Result of different optimization strategies for flying through the four points marked with "+". Red denotes the resulting trajectory from optimizing over the snap, blue for jerk and green for acceleration. The right side shows the trajectories for the x-axis for velocity (top), acceleration (center) and jerk (bottom), again with the same color coding for the different methods.

B. Trajectory Following

In the following experiment, we show how our helicopter can follow smooth trajectories through multiple waypoints. Moreover, the aim here is to also evaluate the effect of optimizing the cost function of the trajectory planner (Eq. (23)) over different derivatives (acceleration, jerk, snap) of the position on the trajectory and the performance of the helicopter. For this, we set 4 waypoints (denoted by "+" in Fig. 7) at the corners of a $1.5\,\text{m} \times 1.5\,\text{m}$ area, which were connected by our trajectory generator. The trajectory starts at 0,0 with zero velocity, goes through the remaining three waypoints without stopping, ending back at point 0,0. We did not specify segment times and left this optimization to the method described in Section IV.

By first inspection of Fig. 7, left, it seems that optimizing Eq. (23) over the acceleration would be a good choice, since it chooses the shortest path of the three methods. However, by looking at the derivatives of the position on the right side, it turns out that the acceleration method tries to save acceleration as much as possible while creating an oscillatory behavior for the jerk. This translates into angular rate (Eq. (4)) and causes ugly motion of the helicopter. In

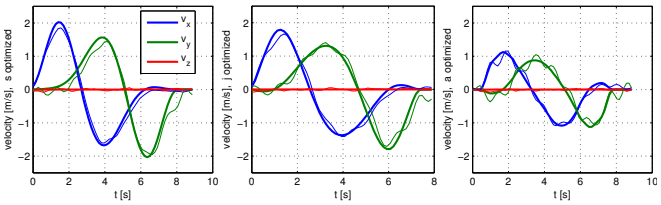

Fig. 8: Velocities of the helicopter flying along the trajectory shown in Fig. 7 From left to right: optimization over snap, jerk and acceleration, where the increasing oscillatory behavior of the jerk becomes more and more visible. In contrast, the speed necessary to travel along the trajectory decreases from left to right.

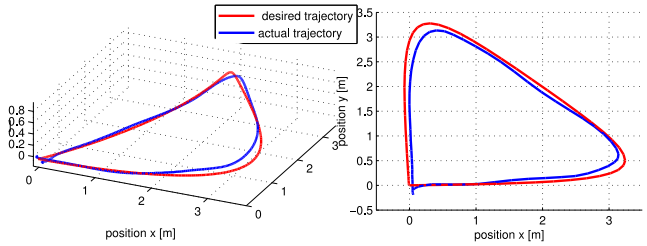

Fig. 9: Fast trajectory flight. The track speed reached up to 3.5 m/s and we obtained a RMS error of 11.8 cm and 7.7 cm, respectively for left and right

contrast, snap shows the smoothest trajectory, but results in a longer path, when no further constraints are specified.

Fig. 8 shows the speed required for flying along the path shown in Fig. 7. The increasingly oscillatory behavior of the jerk, becomes more and more evident when changing the derivative to be optimized from snap to acceleration. However, it should also be noted that the velocity required for the snap to travel along the trajectory in optimal time is twice as high as for the acceleration optimization. Similar observations were made with differently shaped trajectories, where the velocity or its direction changes. For straight line connections, as shown in the first experiment, the methods do not differ much. To summarize, there seems to be no good general answer on which method is to be favored and this might depend on the application. However, optimizing over the jerk seems be a good trade-off. Further investigation of this is necessary and is part of ongoing work.

Lastly, another experiment is shown in Fig. 9. The figure on the left is similar to Fig. 7. On the right, we added the third dimension. The track speed reached up to 3.5 m/s and we obtained a RMS error of 11.8 cm and 7.7 cm, respectively. For fair comparisons to [2] or [5], keep in mind that our helicopter is heavier and thus less agile. Also, we reduced the temporal and spatial accuracy of the Vicon system to simulate realistic visual navigation based flights. Besides the aforementioned experiments, we show a larger scale outdoor experiment in the accompanied video. In this experiment, the helicopter flies a $15 \times 15\,\mathrm{m}$ trajectory in a height of 4 m similar to Fig. 7 while reaching a track speed up to 4 m/s.

VI. Conclusions

In this work we address the issue of position control of a MAV, being able to follow dynamic trajectories. Following a two-loop design based on dynamic inversion, we eliminate the need of the typically used attitude loop, minimizing the computational complexity and improving bandwidth. We show that by using the attitude given by a reliable state estimator, we employ an outer control loop to directly command angular rates and thrust. In turn, as inner control loop, is a simple proportional controller of angular rates. We showed a method for control allocation for a hexacopter, and we presented an approach to estimate vehicle parameters which are tedious to determine. Finally, we successfully demonstrated the whole system in real experiments. Future research involves handling actuator saturation, and improving the generated trajectories for smooth and efficient paths.

VII. Acknowledgments

We like to thank the team from Ascending Technologies for their help and fruitful discussions. We would also like to thank Prof. Raffaello D' Andrea and his team for giving us access to the Flying Machine Arena where we performed our indoor experiments.

References

[1] J. Wang, T. Bierling, L. Höcht, F. Holzapfel, S. Klose, and A. Knoll, "Novel Dynamic Inversion Architecture Design for Quadrocopter Control," in *Conference on Guidance Navigation and Control*, 2011.

[2] D. Mellinger and V. Kumar, " Minimum Snap Trajectory Generation and Control for Quadrotors ," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[3] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous mav," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[4] S. Weiss, "Vision based navigation for micro helicopters," Ph.D. dissertation, ETH Zurich, 2012.

[5] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[6] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," in *Proceedings of the International Federation of Automatic Control (IFAC) world congress*, 2011.

[7] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[8] A. Schoellig, C. Wiltsche, and R. D'Andrea, "Feed-forward parameter identification for precise periodic quadrocopter motions," 2012.

[9] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[10] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *International Journal of Robotics Research (IJRR)*, vol. 30, no. 1, pp. 56–79, 2011.

[11] F. Mirzaei and S. Roumeliotis, "A Kalman Filter-Based Algorithm for IMU-Camera Calibration: Observability Analysis and Performance Evaluation," *IEEE Transactions on Robotics and Automation*, vol. 24, no. 5, pp. 1143 –1156, 2008.

[12] H. Khalil, *Nonlinear Systems*. Macmillan, 1992.

[13] J. Engel, J. Sturm, and D. Cremers, "Camera-Based Navigation of a Low-Cost Quadrocopter," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012.

[14] M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "Path Planning for Motion Dependent State Estimation on Micro Aerial Vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.