# Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor

Matthias Faessler, Flavio Fontana, Christian Forster and Davide Scaramuzza

***Abstract*— Autonomous, vision-based quadrotor flight is widely regarded as a challenging perception and control problem since the accuracy of a flight maneuver is strongly influenced by the quality of the on-board state estimate. In addition, any vision-based state estimator can fail due to the lack of visual information in the scene or due to the loss of feature tracking after an aggressive maneuver. When this happens, the robot should automatically re-initialize the state estimate to maintain its autonomy and, thus, guarantee the safety for itself and the environment. In this paper, we present a system that enables a monocular-vision–based quadrotor to automatically recover from any unknown, initial attitude with significant velocity, such as after loss of visual tracking due to an aggressive maneuver. The recovery procedure consists of multiple stages, in which the quadrotor, first, stabilizes its attitude and altitude, then, re-initializes its visual state-estimation pipeline before stabilizing fully autonomously. To experimentally demonstrate the performance of our system, we aggressively throw the quadrotor in the air by hand and have it recover and stabilize all by itself. We chose this example as it simulates conditions similar to failure recovery during aggressive flight. Our system was able to recover successfully in several hundred throws in both indoor and outdoor environments.**

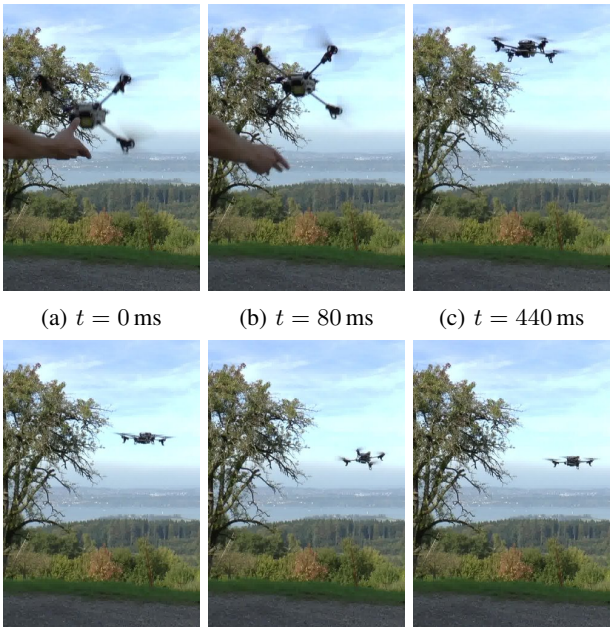
## Supplementary Material

A video attachment to this work is available at: `http://rpg.ifi.uzh.ch/aggressive_flight.html`.

## I. Introduction

### A. Motivation

Autonomous Micro Aerial Vehicles (MAVs) will soon play a major role in remote inspection and search-and-rescue missions. In these applications, the MAVs will have to operate in unknown indoor and outdoor environments, which prevents them from relying on external positioning systems (e.g. GPS or motion-capture systems). A viable solution to maintain position tracking for lightweight MAVs is to use on-board cameras. Unfortunately, vision algorithms are prone to lose visual tracking during fast motions, e.g., when executing aggressive maneuvers, or under strong illumination changes that can occur when transitioning from dark to bright scenes. When visual tracking is lost, the vehicle typically has to descend and land in a partially open-loop maneuver, or a trained pilot has to take over control. To re-initialize the vision pipeline, manual procedures (by hand or remote control) are required by the operators, which renders re-initialization during flight very difficult or even impossible.



(a) $t = 0\,\text{ms}$ (b) $t = 80\,\text{ms}$ (c) $t = 440\,\text{ms}$

(d) $t = 1{,}120\,\text{ms}$ (e) $t = 1{,}640\,\text{ms}$ (f) $t = 2{,}000\,\text{ms}$

Fig. 1: Autonomous recovery after throwing the quadrotor by hand: (a) the quadrotor detects free fall and (b) starts to control its attitude to be horizontal. Once it is horizontal, (c) it first controls its vertical velocity and then, (d) its vertical position. The quadrotor uses its horizontal motion to initialize its visual-inertial state estimation and uses it (e) to first break its horizontal velocity and then (f) lock to the current position.

In this paper, we describe an approach to allow a monocular-vision–based quadrotor to recover completely autonomously from difficult conditions, where it has lost visual tracking, and automatically re-initialize its vision pipeline during flight. This enables the quadrotor to recover in case of a failure of the vision pipeline and continue its mission without landing. Some snapshots of an autonomous recovery outdoors are shown in Fig. 1. When performing aggressive maneuvers, our system allows us to push the quadrotor to its limits and beyond, while still being able to recover at any time without resorting to any external pose-estimation fall-back. Along with re-initializing in flight, our system enables instant launches of the quadrotor by manually throwing it in the air. This starting procedure is not only very quick, but also enables an untrained operator to start the quadrotor without remote control.

The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—`http://rpg.ifi.uzh.ch`. This research was supported by the Swiss National Science Foundation through project number 200021-143607 (Swarm of Flying Cameras) and the National Centre of Competence in Research Robotics (NCCR).

### B. Related Work

In the recent years, several groups have demonstrated MAVs that can perform impressive aerobatics [1], [2], pass through narrow gaps [3], or recover and stabilize virtually from any initial condition [1], [3]. However, besides being limited to a set of learned maneuvers, the platforms used in these demonstrations relied on the accurate and high-frequency position estimates provided by external cameras (such as Vicon[1] or custom-made motion-capture systems) and off-board computation. Nonetheless, these systems need pre-installation and calibration of the cameras and, therefore, cannot be used in unknown and yet-unexplored environments.

To the best of our knowledge, aggressive maneuvers similar to the works mentioned above have not yet been achieved with autonomous quadrotors that rely on on-board sensing and on-board computation, since their state estimate is not as precise and reliable as the one from external positioning systems. To overcome the limitation of being restricted to the volume of a motion-capture system, on-board sensors, such as cameras and Inertial Measurement Units (IMU), are the only viable solution for lightweight MAVs, as demonstrated in [4], [5], [6]. However, current vision-based quadrotors still operate in near-hover conditions, with only few attempts, such as [7], to perform more aggressive maneuvers.

If a quadrotor's vision pipeline fails, there is typically a small set of options left: (i) a pilot must take over; (ii) the quadrotor must land immediately; (iii) the quadrotor must use simple fall backs for stabilization (e.g., based on optical flow algorithms [4]), which do not allow the continuation of its mission without further actions. In [5], a linear sliding window formulation for monocular visual-inertial systems was presented to make a vision-based quadrotor capable of failure recovery and on-the-fly initialization. However, this work assumed that visual features could be extracted and correctly tracked right from the beginning of the recovery procedure.

Along with possible failures of their state-estimation pipeline, monocular-vision–based quadrotors present the drawback that they typically require an initialization phase before they can fly autonomously. This initialization is usually performed by moving the quadrotor by hand or through remote control. Since this is time consuming and not easy to perform, attempts have been made to perform the initialization automatically. For instance, in [8], the authors presented a system that allows the user to toss a quadrotor in the air, where it then initializes a visual-odometry pipeline. Nevertheless, that system still required several seconds for the state estimate to converge before the toss and several more seconds until the visual-odometry pipeline was initialized. A closed-form solution for state estimation with a visual-inertial system that does not require initialization was presented in [9]. However, this approach is not suitable for systems that rely on noisy sensor data.

[1] www.vicon.com

### C. Contributions and Outline

We present a system that enables a monocular vision-based quadrotor to autonomously recover from state-estimation failures quickly, and re-initialize its visual-inertial state estimation. The described system allows the quadrotor to recover from any attitude, even with high linear velocities and body rates. The performance of our recovery strategy is evaluated in the scenario where a quadrotor is thrown in the air by hand and must stabilize based only on its on-board sensors. We present an attitude estimator based on quaternions which fuses measurements from the gyroscopes and accelerometers to obtain a globally-valid attitude estimate at the time when it is launched. This allows the user to throw the quadrotor with any initial attitude, and the controller immediately starts to guide it to horizontal position. In contrast to [5], our system does not require the observation of visual features at the beginning of the recovery procedure but only once its attitude is stabilized, which simplifies feature tracking greatly and reduces computational complexity. In addition to [8], no preparation time before launching the quadrotor is required and the entire recovery is performed more quickly.

Along with a very quick and easy start by simply throwing the quadrotor in the air, our system enables more aggressive flight, where a vision-based quadrotor can perform a maneuver with the expectation that it will lose tracking but still regain control.

The remainder of this paper is organized as follows. Section II describes the implemented high-level and low-level controllers, as well as estimation algorithms used for recovery. Section III describes the different stages of our recovery procedure. Section IV introduces our quadrotor platform and presents the experimental results. Finally, Section V concludes the work.

## II. Control and State Estimation

This section describes our control and state estimation algorithms that are used for recovery as well as for vision-based flying with our quadrotor. The controller is split into a high-level part and a low-level part. The high-level controller enables the quadrotor to track desired positions and velocities, whereas the low-level controller enables it to track desired attitudes or body rates. The overall state estimation and control structure with the used sensors is illustrated in Fig. 2.

### A. Nomenclature

When describing the control and state estimation, we make use of some notation that we introduce here for clarity. We use a hat (e.g. $\hat{v}$) and a tilde (e.g. $\tilde{v}$) to denote an estimated and measured value, respectively. To describe the multiplication of two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ we write $\mathbf{q}_1 \otimes \mathbf{q}_2$, and we write $\mathbf{q} \odot \mathbf{v}$ for the rotation of a vector $\mathbf{v}$ by the quaternion $\mathbf{q}$. Furthermore, we express the basis vectors of a coordinate system as e.g. $\mathbf{e}_x^B$ which denotes the $x$-basis vector of the body coordinate system, as illustrated in Fig. 3. A prescript (e.g. ${}_B\mathbf{v}$) indicates that the vector $\mathbf{v}$ is expressed in the body coordinate system. Vectors without prescripts are

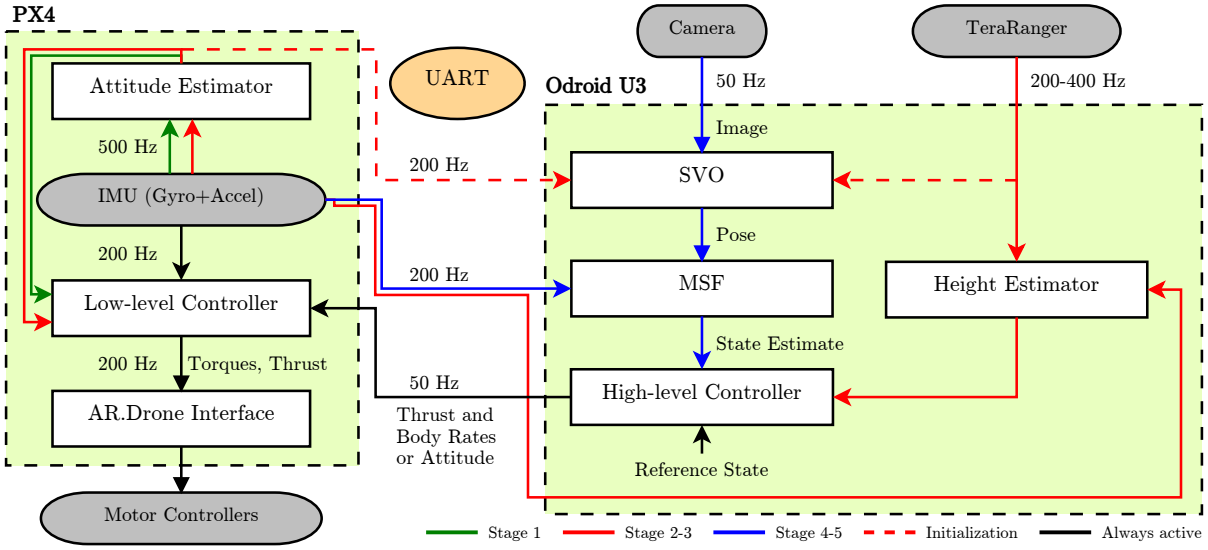



Fig. 2: Control-system overview: the PX4 and Odroid U3 communicate with each other over a UART interface. Gray boxes are sensors and actuators; white boxes depict software modules. Green arrows indicate the communication that is required specifically during stage 1, red arrows indicate the communication that is required specifically during stages 2 and 3, and blue arrows indicate the communication that is required during stages 4 and 5 as well as for normal vision-based flight. The dashed red arrows indicate measurements that are used only once for initializing SVO. Black arrows indicate communication that is required in all the recovery stages as well as for normal vision-based flight. A more detailed description of the hardware set-up is given in Section IV-A

expressed in the world coordinate system with the exception of the body rates $\boldsymbol{\omega}$, which are always expressed in body coordinates.

### B. Dynamical Model

For state estimation and control, we make use of the following dynamical model for our quadrotor:

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{1}$$
$$\dot{\mathbf{v}} = \mathbf{g} + \mathbf{q} \odot \mathbf{c}, \tag{2}$$
$$\dot{\mathbf{q}} = \boldsymbol{\Lambda}(\boldsymbol{\omega}) \cdot \mathbf{q}, \tag{3}$$
$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} \cdot (\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}), \tag{4}$$

where $\mathbf{r} = [x\ y\ z]^T$ and $\mathbf{v} = [v_x\ v_y\ v_z]^T$ are the position and velocity in world coordinates, $\mathbf{q} = [q_w\ q_x\ q_y\ q_z]^T$ is the orientation of the quadrotor's body coordinates with respect to the world coordinates, and $\boldsymbol{\omega} = [p\ q\ r]^T$ denotes the body rates (roll, pitch and yaw, respectively) expressed in body coordinates. The skew-symmetric matrix $\boldsymbol{\Lambda}(\boldsymbol{\omega})$ is defined as

$$\boldsymbol{\Lambda}(\boldsymbol{\omega}) = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}. \tag{5}$$

We define the gravity vector as $\mathbf{g} = [0\ 0\ -g]^T$ with $g = 9.81\,\mathrm{m\,s^{-2}}$ and the second-order moment-of-inertia matrix of the quadrotor as $\mathbf{J} = \mathrm{diag}\,(J_{xx},\ J_{yy},\ J_{zz})$. The mass-normalized thrust vector is $\mathbf{c} = [0\ 0\ c]^T$, with

$$mc = f_1 + f_2 + f_3 + f_4, \tag{6}$$

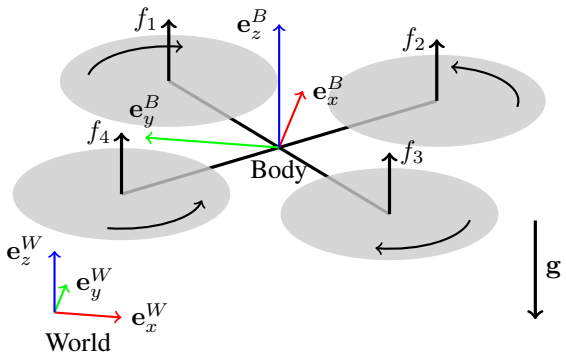



Fig. 3: Quadrotor with coordinate system and rotor forces.

where $m$ is the mass of the quadrotor and $f_i$ are the four motor thrusts as illustrated in Fig. 3. The torque inputs $\boldsymbol{\tau}$ are composed of the single-rotor thrusts as

$$\boldsymbol{\tau} = \begin{bmatrix} \frac{\sqrt{2}}{2} l (f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2} l (-f_1 - f_2 + f_3 + f_4) \\ \kappa (f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \tag{7}$$

where $l$ is the quadrotor arm length and $\kappa$ is the rotor-torque coefficient.

Our coordinate-system conventions and rotor numbering are illustrated in Fig 3.

### C. High-Level Controller

The high-level controller takes a reference state and a state estimate as input and computes the desired attitude or desired body rates, which are then sent to the low-level controller. A reference state consists of a reference position $\mathbf{r}_{ref}$, a

reference velocity $\mathbf{v}_{ref}$, a reference acceleration $\mathbf{a}_{ref}$, and a reference heading $\psi_{ref}$. First, we describe the position controller, followed by the attitude controller.

*1) Position Controller:* To track a reference trajectory, we implemented a PD controller with feed-forward terms on the reference acceleration from the trajectory and gravity:

$$\mathbf{a}_{des} = \mathbf{P}_{pos} \cdot (\mathbf{r}_{ref} - \hat{\mathbf{r}}) + \mathbf{D}_{pos} \cdot (\mathbf{v}_{ref} - \hat{\mathbf{v}}) + \mathbf{a}_{ref} - \mathbf{g}, \quad (8)$$

with gain matrices $\mathbf{P}_{pos} = \mathrm{diag}\,(p_{xy}, p_{xy}, p_z)$ and $\mathbf{D}_{pos} = \mathrm{diag}\,(d_{xy}, d_{xy}, d_z)$. To compute the desired normalized thrust $c_{des}$, we project the desired acceleration onto the current ${}_W\mathbf{e}_z^B$ axis

$$c_{des} = \mathbf{a}_{des} \cdot \mathbf{e}_z^B. \quad (9)$$

The output of the position controller is the desired accelerations $\mathbf{a}_{des}$. The desired acceleration, together with the reference heading $\psi_{ref}$, encodes the desired orientation as well as a mass normalized thrust $c_{des}$.

*2) Attitude Controller:* Since a quadrotor can only accelerate in its body $z$ direction, $\mathbf{a}_{des}$ enforces two degrees of freedom of the desired attitude. The third degree of freedom is enforced by the reference heading $\psi_{ref}$. Note that the rotation around the body $z$ axis has no influence on the translational behaviour of the quadrotor. Therefore, we want to align the body $z$ axis with the desired acceleration $\mathbf{a}_{des}$ by rotating around the body $x$ and $y$ axes and use rotations around the body $z$ axis only to control the heading. Our quadrotors have much more attitude control authority on the $x$ and $y$ axes than on the $z$ axis because there, they can make use of thrust differences as opposed to differences in rotor drag torques. The moment due to the maximum-possible thrust difference is much larger than the maximum possible difference of rotor drag torques. For this reason, we separate the attitude control into two parts as described in the following.

*a) Desired Roll and Pitch Rates:* From the current attitude estimate and the desired acceleration, we can compute the current and the desired body $z$ axes, respectively, as

$$\hat{\mathbf{e}}_z^B = \hat{\mathbf{q}} \otimes [0\ 0\ 1]^T, \quad \mathbf{e}_{z,des}^B = \frac{\mathbf{a}_{des}}{\|\mathbf{a}_{des}\|}. \quad (10)$$

Now, we design an error quaternion that describes the necessary rotation to align these two vectors. To do so, we compute the angle $\alpha$ between the two vectors and a normal vector $\mathbf{n}$ to both of them:

$$\alpha = \arccos(\hat{\mathbf{e}}_z^B \cdot \mathbf{e}_{z,des}^B), \quad (11)$$

$$\mathbf{n} = \frac{\hat{\mathbf{e}}_z^B \times \mathbf{e}_{z,des}^B}{\left\|\hat{\mathbf{e}}_z^B \times \mathbf{e}_{z,des}^B\right\|}. \quad (12)$$

Since we want to apply this rotation with respect to the current body orientation, we have to transform the rotation axis $\mathbf{n}$ into body coordinates using the current attitude estimate $\hat{\mathbf{q}}$

$${}_B\mathbf{n} = \hat{\mathbf{q}}^{-1} \odot \mathbf{n}. \quad (13)$$

The error quaternion can then be constructed as

$$\mathbf{q}_{e,rp} = \begin{bmatrix} \cos(\frac{\alpha}{2}) \\ {}_B\mathbf{n} \sin(\frac{\alpha}{2}) \end{bmatrix}. \quad (14)$$

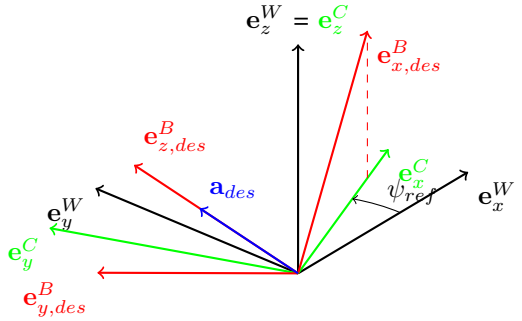



Fig. 4: Coordinate frames used in the attitude controller. We make use of the coordinate frame $C$, which is obtained by rotating the world frame ($W$) by the reference heading $\psi_{ref}$ to construct, together with the desired acceleration $\mathbf{a}_{des}$, the full desired orientation of the body frame ($B$).

Note that if $\alpha = 0$, the rotation axis is undetermined and we set the error quaternion $\mathbf{q}_{e,rp}$ to be the identity directly. Also note that by construction, the $z$ component of $\mathbf{q}_{e,rp}$ is always zero, which assures that no rotation around the body $z$ axis is necessary to align $\hat{\mathbf{e}}_z^B$ with $\mathbf{e}_{z,des}^B$. From the error quaternion $\mathbf{q}_{e,rp}$, we can then compute the desired roll and pitch rates with the following control law:

$$\begin{bmatrix} p \\ q \end{bmatrix}_{des} = \begin{cases} 2 \cdot p_{rp} \cdot \mathbf{q}_{e,rp}^{(x,y)} & \text{if } \mathbf{q}_{e,rp}^{(w)} \geq 0 \\ -2 \cdot p_{rp} \cdot \mathbf{q}_{e,rp}^{(x,y)} & \text{if } \mathbf{q}_{e,rp}^{(w)} < 0 \end{cases}. \quad (15)$$

It can be shown that this control law is globally asymptotically stable and its discrete implementation is robust to measurement noise [10], [11].

*b) Desired Yaw Rate:* To compute the desired yaw rate $r_{des}$, we look at the heading error that remains after aligning $\hat{\mathbf{e}}_z^B$ with $\mathbf{e}_{z,des}^B$ with the above control law. To do so, we first compute the full desired attitude. For this, we make use of an intermediate coordinate system $C$, which is the world frame rotated around its $z$ axis by the desired heading $\psi_{ref}$ as illustrated in Fig. 4. The $x$ and $y$ axes of the coordinate system $C$ are defined as

$$\mathbf{e}_x^C = [\cos(\psi_{ref})\ \sin(\psi_{ref})\ 0]^T, \quad (16)$$

$$\mathbf{e}_y^C = [-\sin(\psi_{ref})\ \cos(\psi_{ref})\ 0]^T. \quad (17)$$

The goal of rotating around the quadrotor's $z$ axis is to align the projection of its $x$ axis onto the world $x-y$ plane with $\mathbf{e}_x^C$. This forces the desired body $x$ axis $\mathbf{e}_{x,des}^B$ to lie in a plane spanned by $\mathbf{e}_x^C$ and $\mathbf{e}_z^W$, which is fulfilled by

$$\mathbf{e}_{x,des}^B = \frac{\mathbf{e}_y^C \times \mathbf{e}_{z,des}^B}{\left\|\mathbf{e}_y^C \times \mathbf{e}_{z,des}^B\right\|}. \quad (18)$$

Note that if this cross product is zero, there are infinitely many rotations around the desired body $z$ axis that achieve the desired heading. Therefore, in that case, we apply a desired yaw rate $r_{des} = 0$.

If the desired body $z$ axis has a negative $z$ component (i.e. $\mathbf{e}_{z,des}^B$ is pointing downwards), the projection of the computed desired body $x$ axis into the horizontal plane will point in the opposite direction of $\mathbf{e}_x^C$, therefore we use the

negation of it. From $\mathbf{e}^B_{x,des}$ and $\mathbf{e}^B_{z,des}$, we can then compute $\mathbf{e}^B_{y,des}$ as

$$\mathbf{e}^B_{y,des} = \frac{\mathbf{e}^B_{z,des} \times \mathbf{e}^B_{x,des}}{\left\| \mathbf{e}^B_{z,des} \times \mathbf{e}^B_{x,des} \right\|}. \tag{19}$$

Now, the full desired attitude $\mathbf{q}_{des}$ can be built from the three desired body axes $\mathbf{e}^B_{x,des}$, $\mathbf{e}^B_{y,des}$ and $\mathbf{e}^B_{z,des}$. Our definition of the heading is different than in the controller presented in [10] and has the advantage of being meaningful for any orientation of the quadrotor, which is not the case, for instance, when using a definition based on Euler angles. Now we can compute an error quaternion that describes the necessary rotation to achieve the full desired attitude after rotating by $\mathbf{q}_{e,rp}$ as

$$\mathbf{q}_{e,y} = (\hat{\mathbf{q}} \otimes \mathbf{q}_{e,rp})^{-1} \otimes \mathbf{q}_{des}. \tag{20}$$

Note that the $x$ and $y$ components of $\mathbf{q}_{e,y}$ are always zero. Similarly to (15), we can now compute the desired yaw rate $r_{des}$ from $\mathbf{q}^{(z)}_{e,y}$ with a gain $p_{yaw}$. Splitting the attitude control into these two parts allows us to have different gains for $p_{rp}$ and $p_{yaw}$, which is desirable due to different control limits.

### D. Low-Level Controller

The commands sent to the low-level controller on the PX4 are the desired body rates $\boldsymbol{\omega}_{des}$ and the desired mass-normalized thrust $c_{des}$. From the desired body rates $\boldsymbol{\omega}_{des}$ and the measured body rates $\tilde{\boldsymbol{\omega}}$, we can compute desired torques $\boldsymbol{\tau}_{des}$ with a feedback linearizing control scheme:

$$\boldsymbol{\tau}_{des} = \mathbf{J} \cdot \mathbf{P}_{att} \cdot (\boldsymbol{\omega}_{des} - \hat{\boldsymbol{\omega}}) + \hat{\boldsymbol{\omega}} \times \mathbf{J}\hat{\boldsymbol{\omega}}, \tag{21}$$

where $\mathbf{P}_{att} = \operatorname{diag}(p_{pq}, p_{pq}, p_r)$. Then, we can substitute $\boldsymbol{\tau}_{des}$ and $c_{des}$ into (7) and (6) and solve them for the desired rotor thrusts that must be applied.

### E. IMU-Based Attitude Estimation

In recovering after a loss of visual tracking, or when launching a quadrotor by hand, we need to have an attitude estimate available for controlling the attitude until the vision pipeline is initialized and running. We achieve this by using a quaternion-based attitude state estimator that fuses the measurements of the gyroscopes and accelerometers at $500\,\mathrm{Hz}$. The implemented attitude estimator works in a prediction-update scheme where the prediction is performed based on the gyroscope measurements and the update step is performed based on the accelerometer measurements.

We predict the attitude estimate at the time of the current IMU measurement from the previous attitude estimate by integrating the gyroscope measurements over a time $\Delta t$ between the previous and the current IMU measurement. This integration is performed with a zero-th order quaternion integration, as described in [12], assuming that the body rates are constant over a time $\Delta t$

$$\hat{\mathbf{q}}_{pred}(k) = \left( \mathbf{I}_4 \cdot \cos\left( \frac{\|\hat{\boldsymbol{\omega}}\| \, \Delta t}{2} \right) + \frac{2}{\|\hat{\boldsymbol{\omega}}\|} \cdot \boldsymbol{\Lambda}(\hat{\boldsymbol{\omega}}) \cdot \sin\left( \frac{\|\hat{\boldsymbol{\omega}}\| \, \Delta t}{2} \right) \right) \cdot \hat{\mathbf{q}}(k-1), \tag{22}$$

where $\mathbf{I}_4 \in \mathbb{R}^{4\times4}$ denotes the identity matrix. We chose a zero-th order integration since it has a performance similar to a first order integration, but at a lower computational load. Note that (22) can only be evaluated if $\|\hat{\boldsymbol{\omega}}\| \neq 0$, otherwise we keep the attitude estimate prediction constant $\hat{\mathbf{q}}_{pred}(k) = \hat{\mathbf{q}}(k-1)$.

As long as the quadrotor is hand held, we assume that the accelerometers are measuring $\tilde{\mathbf{a}} = -\mathbf{g}$ on average, which gives us information about the gravity direction. Therefore, we correct the predicted attitude estimate such that the corresponding body $z$ direction $\hat{\mathbf{e}}^B_{z,pred}$ rotates towards the measured acceleration $\tilde{\mathbf{a}}$. To do so, we compute the angle and axis of rotation required to align $\hat{\mathbf{e}}^B_{z,pred}$ with $\tilde{\mathbf{a}}$ as

$$\beta = \arccos\left( {}_B\hat{\mathbf{e}}^B_{z,pred} \cdot \frac{\tilde{\mathbf{a}}}{\|\tilde{\mathbf{a}}\|} \right), \tag{23}$$

$${}_B\mathbf{h} = \frac{{}_B\tilde{\mathbf{a}} \times_B \hat{\mathbf{e}}^B_{z,pred}}{\left\| {}_B\tilde{\mathbf{a}} \times_B \hat{\mathbf{e}}^B_{z,pred} \right\|}. \tag{24}$$

Since the measured acceleration is noisy, we weigh the angle $\beta$ by a gain $k_{corr} < 1$ and use it to design the following correction quaternion:

$$\mathbf{q}_{corr} = \begin{bmatrix} \cos(k_{corr} \cdot \frac{\beta}{2}) \\ {}_B\mathbf{h} \sin(k_{corr} \cdot \frac{\beta}{2}) \end{bmatrix}, \tag{25}$$

which we apply to the predicted attitude estimate to get the corrected attitude estimate at the current time

$$\hat{\mathbf{q}}(k) = \hat{\mathbf{q}}_{pred}(k) \otimes \mathbf{q}_{corr}. \tag{26}$$

Note that the update step is only performed if the measured body rates are small, i.e. $\|\tilde{\boldsymbol{\omega}}\| < 0.5\,\mathrm{rad\,s^{-1}}$, and the magnitude of the measured acceleration is close to the magnitude of the gravitational acceleration, i.e. $|\|\tilde{\mathbf{a}}\| - g| < 1.0\,\mathrm{m\,s^{-2}}$. The first acceleration measurement that meets these criteria is used to get an initial estimate of the gravity direction.

Since this attitude estimator is based on quaternions, it is free of singularities, which is necessary because we want to be able to recover from any initial attitude. It is furthermore based on the assumption that the accelerometers are measuring $\tilde{\mathbf{a}} \approx -\mathbf{g}$ on average, which is valid when the quadrotor is hand held but not when it is flying freely. Still, in near-hover conditions, the attitude estimator does not drift in the roll and pitch estimates (see Fig. 8) because of aerodynamic effects as described in [13].

### F. Height Estimation

To estimate the vertical position $z$ and velocity $\dot{z}$ of the quadrotor, we use a *TeraRanger One* sensor and fuse its measurements with the acceleration measurements in a fixed-gain Kalman filter. The prediction step is performed as

$$\begin{bmatrix} z_{prior} \\ \dot{z}_{prior} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{z}_k \\ \hat{\dot{z}}_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \cdot \tilde{\ddot{z}}_{imu}, \tag{27}$$

and the update step is performed using a fixed gain $\mathbf{K}$ as

$$\begin{bmatrix} \hat{z}_{k+1} \\ \hat{\dot{z}}_{k+1} \end{bmatrix} = \begin{bmatrix} z_{prior} \\ \dot{z}_{prior} \end{bmatrix} + \mathbf{K} \cdot (\tilde{z} - \hat{z}_{prior}). \tag{28}$$

## III. Recovery and Automatic Initialization

In this section, we describe the procedure that our quadrotor executes to recover from a failure of the state estimation pipeline or a manual throw. The recovery procedure is divided into five sequential stages. We describe the control algorithm for each stage and explain the conditions that must be met before advancing to the next stage. These conditions are chosen such that the respective controller can also satisfy them during the following stages. Each stage makes use of the controller described in Section II but with different gains. The duration of each stage depends on how long it takes the controller to meet the conditions to advance to the next stage. The scheme of five stages allow to recover from various conditions after a manual throw or after a failure in the state estimation where stages 1, 2 and 4 might be skipped entirely when the conditions for the subsequent stage are already satisfied.

For our physical platform, we consider a quadrotor equipped with a monocular visual-inertial system consisting of a single camera, an IMU, and a down-looking distance sensor as described further in Section IV-A. We demonstrate our recovery procedure on the scenario where the quadrotor is thrown in the air by hand, and automatically initializes its vision pipeline such that it can control its position purely based on a vision-based state estimate.

### A. Launch Detection

As a first step to recover after tossing the quadrotor in the air, it needs to detect the launch for which it uses its accelerometers. Ideally, without disturbances and noise, the accelerometers measure $\tilde{\mathbf{a}} = -\mathbf{g}$ when standing still (e.g. on the ground) and $\tilde{\mathbf{a}} = \mathbf{0}$ when the quadrotor is in free fall. When in flight, the accelerometers ideally measure just the accelerations due to the applied rotor thrust, i.e. $\tilde{\mathbf{a}} = \mathbf{c}$. Hence, when the quadrotor is launched, we can detect a drop in the measured accelerations to a value corresponding to the currently applied thrust. Therefore, we start recovering when we measure

$$\|\tilde{a}\|_{mean} < \|c_{idle}\| + t_{launch}, \tag{29}$$

where $\|\tilde{a}\|_{mean}$ is the norm of the measured acceleration averaged over the last $50\,\mathrm{ms}$, $c_{idle}$ is the mass-normalized idle thrust that prevent the rotors from standing still and $t_{launch} = 2.0\,\mathrm{m\,s^{-2}}$ is a threshold parameter.

### B. Recovery and Initialization Steps

*1) Attitude Control:* Immediately after a launch is detected, the quadrotor starts to control its orientation to be horizontal. To do so, we use the attitude controller described in Section II-C together with an IMU based estimate of the attitude as described in Section II-E. Since we have no information on height and vertical velocity at this stage, we apply a fixed mass-normalized thrust equal to the gravitational acceleration $c = g$.

As soon as the distance sensor is oriented towards the ground, i.e. the angle between the body $z$ axis $\mathbf{e}_z^B$ and the world $z$ axis $\mathbf{e}_z^W$ (see Fig. 3) is smaller than $20°$ and the roll and pitch rates are small, i.e. $\|\hat{\boldsymbol{\omega}}^{(x,y)}\| < 10\,\mathrm{rad\,s^{-1}}$, we initialize the height filter and switch to the next stage.

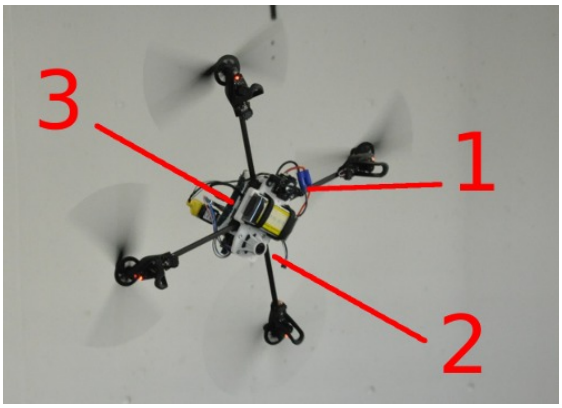



Fig. 5: A closeup of our quadrotor during recovery: 1) TeraRanger One distance sensor, 2) down-looking camera, 3) on-board electronics consisting of an Odroid U3 quad-core computer and a PX4FMU autopilot.

*2) Attitude and Vertical Velocity Control:* Once the distance sensor is pointing towards the ground, we control the horizontal velocity to zero using our position controller (8) but set the proportional gain $\mathbf{P}_{pos}$ and the vertical velocity gain $d_{xy}$ to zero. The vertical velocity is estimated from the distance sensor and the IMU as described in Section II-F. As soon as the vertical velocity is small enough, i.e. $\|\dot{z}\| < 0.3\,\mathrm{m\,s^{-1}}$, we set the current height as the reference height and switch to the next stage.

*3) Attitude and Height Control:* Once the height controller is active, we stabilize the height relative to the ground, together with the attitude. Again, we make use of our position controller (8) but now only set $p_{xy}$ and $d_{xy}$ to zero. At this stage, due to the lack of horizontal position information, the quadrotor drifts in a horizontal plane. We use this horizontal translation to initialize the vision-based state-estimation pipeline with an initial scale corresponding to the current height estimate. After it is initialized, we switch to the next stage.

*4) Velocity Control:* At this point, the quadrotor can still have large horizontal velocities, which we want to lower before locking to the current position. In this stage, we use the position controller (8) but set the proportional gain $p_{xy}$ to zero. Once the quadrotor reaches a small horizontal velocity, i.e. $\|\mathbf{v}^{(x,y)}\| < 0.2\,\mathrm{m\,s^{-1}}$, we lock to the current position and heading.

*5) Position Control:* In this last stage, we lock the quadrotor to the previously specified reference position until the quadrotor is given a new mission. For controlling the quadrotor to its reference position, we use the full control pipeline as described in Section II-C.

## IV. Experiments

### A. Quadrotor Platform

We built our quadrotors from selected off-the-shelf components and custom 3D printed parts (see Fig. 5). They rely on the frame of the Parrot AR.Drone 2.0 including

their motors, motor controllers, gears, and propellers. The platform is powered by one 1,350 mA h LiPo battery which allows a flight time of approximately 10 min.

We completely replaced the electronic parts of the AR.Drone by a PX4FMU autopilot and a PX4IOAR adapter board [14]. The PX4FMU consists, among other components, of an IMU and a micro controller to read the sensors, run the low-level control (21), and command the motors. In addition to the PX4 autopilot, our quadrotors are equipped with an Odroid-U3 single-board computer. It contains a 1.7 GHz quad-core processor (used in Samsung smart phones) running Ubuntu 14.04 and ROS. The PX4 micro controller communicates with the Odroid board over UART (see Fig. 2).

To stabilize the quadrotor, we make use of the gyros and accelerometers of the IMU on the PX4FMU, a downward-looking MatrixVision mvBlueFOX-MLC200w 752 × 480-pixel monochrome camera as well as a downward-looking *TeraRanger One* distance sensor. The *TeraRanger One* is an infra-red range sensor that uses time of flight to measure distances of up to 14 m with up to 2 cm accuracy and a frequency of up to 1 kHz. It works in both indoor and outdoor environments.

The images from the camera are processed on the Odroid by means of our Semi-Direct Visual Odometry (SVO[2]) pipeline [15]. The visual-odometry pipeline outputs an unscaled pose that is then fused with the IMU readings in an Extended Kalman Filter framework (Multi Sensor Fusion (MSF) [16]) to compute a metric state estimate. More details about our quadrotor system are given in [17]

### B. Indoor Experiments

To show the performance of the proposed recovery procedure, we throw a quadrotor by hand in an OptiTrack motion-capture system that we use for ground-truth comparison. For recovery, the quadrotor only uses its on-board sensors and performs all the necessary computations on board. Fig. 6, 7, and 8 show the on-board state estimates compared to OptiTrack measurements. The on-board state estimates are aligned to the OptiTrack data by a single data point each. The vertical dashed lines indicate the start of the five recovery stages with corresponding numbers as described in Section III-B. The individual state estimates are plotted from the time on where they are used for feedback control. At the point where a state estimate becomes available, it is directly used for feedback control.

Fig. 6 shows the height estimate from fusing TeraRanger and IMU measurements, as well as the height estimate from the visual pipeline consisting of SVO and MSF compared to the ground truth height from OptiTrack. The height estimate from fusing the TeraRanger with the IMU is used in stage 3 and 4 and the height estimate from MSF is used in stage 5 for feedback control.

Fig. 7 shows the linear velocity estimates from MSF compared to velocity estimates from OptiTrack. In the vertical direction, we also show the velocity estimate from fusing TeraRanger and IMU measurements. This velocity estimate is used during stages 2 and 3. During stages 4 and 5, the velocity estimates from MSF are used for feedback control.

Fig. 8 shows the roll and pitch angles from the attitude estimate purely based on the IMU measurements and from the MSF compared to the orientation measurements from OptiTrack. The IMU-based attitude estimate is used for control during stages 1, 2, and 3. During stages 4 and 5, the attitude estimate from MSF is used for feedback control. In the IMU-based attitude estimator, unit quaternions are used to represent the quadrotor's attitude and they are only transformed into Euler Angles for visualization. Note that the roll and pitch estimates from the IMU-based attitude estimator do not drift even beyond recovery.

[2] http://github.com/uzh-rpg/rpg_svo

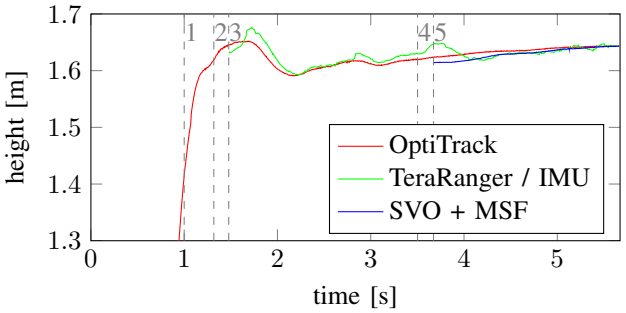



Fig. 6: Height estimates from TeraRanger / IMU fusion and the vision pipeline (SVO + MSF) compared to ground truth from OptiTrack. The vertical lines correspond to the start of each recovery stage as described in Section III-B.

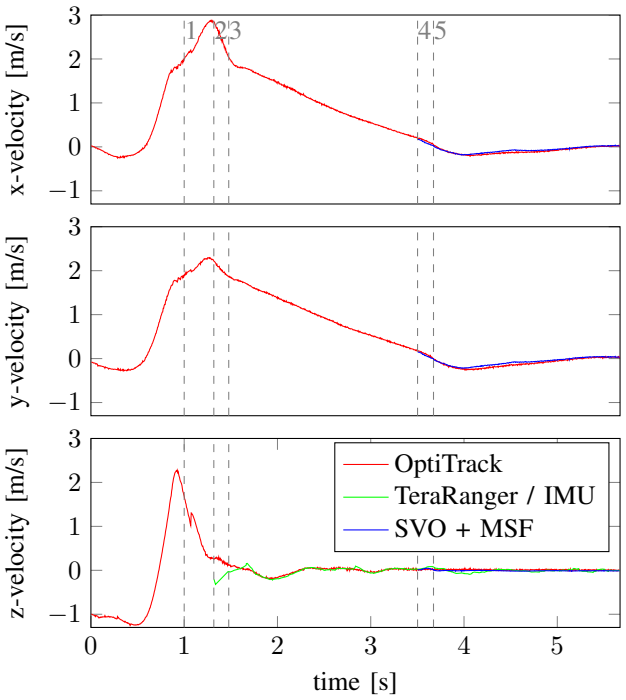



Fig. 7: Velocity estimates from the vision pipeline (SVO + MSF) and from the TeraRanger / IMU fusion compared to ground truth from OptiTrack.

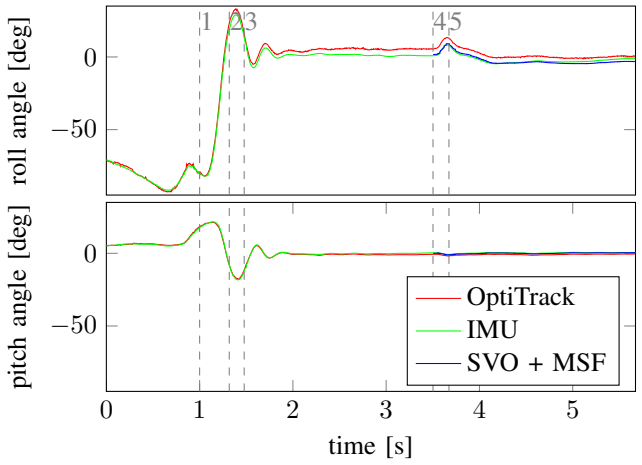



Fig. 8: Roll and Pitch estimates from the IMU-based attitude estimator (see Section II-E) and the vision pipeline (SVO + MSF) compared to ground truth from OptiTrack.

We performed several hundred throws indoors with maximum linear accelerations above $25\,\mathrm{m\,s^{-2}}$, maximum body rates above $650\,^\circ\,\mathrm{s^{-1}}$, and maximum linear velocities exceeding $3.6\,\mathrm{m\,s^{-1}}$ with successful recoveries. We reached a success rate of more than $85\,\%$, where failures occurred when the vision pipeline could not initialize properly, e.g., when flying close to the ground with high velocities. Examples from indoor recoveries are given in the enclosed video.

### C. Outdoor Experiments

We used the same set-up as for the indoor experiments to throw the quadrotor outdoors and have it recover. Because of the absence of ground truth outdoors, plots of state estimates are not shown. In more than 30 throws with successful recovery, the quadrotor reached maximum linear accelerations above $40\,\mathrm{m\,s^{-2}}$, maximum body rates above $800\,^\circ\,\mathrm{s^{-1}}$, and maximum linear velocities above $6\,\mathrm{m\,s^{-1}}$. Examples from outdoor recoveries are given in Fig. 1 and in the enclosed video. We achieved a similar success rate as for the indoor experiments where failures are additionally caused by disturbances of the TeraRanger due to sun light.

## V. Conclusion

We developed a system that enables a monocular-vision-based quadrotor to recover and re-initialize its vision-based state estimation pipeline from any attitude, even with significant initial linear velocities. The on-board system receives the measurements from an IMU, a single camera, and a range sensor and fuses this information to stabilize the quadrotor by means of a cascaded control structure. We designed a recovery procedure consisting of five sequential stages with several controller types: purely inertial, range-inertial, and visual-inertial. To demonstrate its capabilities, we threw the quadrotor in the air by hand and had it recover autonomously. In contrast to existing work, the proposed system does not need any initialization before the quadrotor is launched. In indoor experiments, the state estimates obtained by our system agree well with those measured by a motion capture system. In indoor and outdoor experiments, we demonstrated that the quadrotor successfully decelerates and stabilizes within approximately two to three seconds after throwing it aggressively in the air. Our quadrotor was able to recover successfully in several hundred throws in both unknown indoor and outdoor environments with a success rate of more than $85\,\%$. Our system not only allows instant launches but also enables mid-air re-initialization after aggressive open-loop maneuvers or in case visual tracking is lost.

## References


[1] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *Int. J. Rob. Res.*, vol. 29, no. 13, pp. 1608–1639, Nov. 2010.

[2] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2010, pp. 1642–1648.

[3] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *Intl. Sym. on Experimental Robotics (ISER)*, Dec 2010.

[4] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2012, pp. 957–964.

[5] S. Shen, "Autonomous navigation in complex indoor and outdoor environments with micro aerial vehicles," Ph.D. dissertation, University of Pennsylvania, Philadelphia, PA, USA, July 2014.

[6] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. B. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robotics Automation Magazine*, 2014.

[7] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards aggressive flight with a quadrotor," in *Robotics: Science and Systems (RSS)*, June 2013.

[8] R. Brockers, M. Hummenberger, S. Weiss, and L. Matthies, "Towards autonomous navigation of miniature UAV," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 631–637.

[9] A. Martinelli, "Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 44–60, Feb. 2012.

[10] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control," Department of Mechanical and Process Engineering, ETHZ, Tech. Rep., Oct. 2013.

[11] C. Mayhew, R. Sanfelice, and A. Teel, "Quaternion-based hybrid control for robust global attitude tracking," *Automatic Control, IEEE Transactions on*, vol. 56, no. 11, pp. 2555–2566, Nov 2011.

[12] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep. 2005-002, Mar. 2005.

[13] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sept 2012.

[14] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1–2, pp. 21–39, 2012.

[15] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[16] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

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