2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments

Markus Achtelik, Michael Achtelik, Stephan Weiss, Roland Siegwart

Abstract— **In this paper, we present our latest achievements towards the goal of autonomous flights of an MAV in unknown environments, only having a monocular camera as exteroceptive sensor. As MAVs are highly agile, it is not sufficient to directly use the visual input for position control at the framerates that can be achieved with small onboard computers. Our contributions in this work are twofold. First, we present a solution to overcome the issue of having a low frequent onboard visual pose update versus the high agility of an MAV. This is solved by filtering visual information with inputs from inertial sensors. Second, as our system is based on monocular vision, we present a solution to estimate the metric visual scale aid of an air pressure sensor. All computation is running onboard and is tightly integrated on the MAV to avoid jitter and latencies. This framework enables stable flights indoors and outdoors even under windy conditions.**

I. Introduction

The research in autonomous micro helicopters is advancing and evolving fast. Even though a lot of progress has been achieved in this topic during the past years, the community is still striving to achieve simple autonomous flights in unknown and GPS denied environments. Only after solving this issue, high level tasks such as autonomous exploration, swarming, and large trajectory planning can be tackled.

Stable flights and navigation with GPS are well explored and work out of the box [1]. However, GPS is not a reliable service as its availability can be limited by urban canyons and is completely unavailable in indoor environments. The alternative of using laser range finders is not optimal since these sensors have a restricted perception distance and are still heavy for MAVs.

Considering the above mentioned and to be independent of the (quality of the) GPS signal, a viable solution is to navigate with a vision based system. This ensures operations of the MAV indoors as well as outdoors. Recently we presented, to the best of our knowledge, the first vision based solution for completely autonomous flights in unknown and GPS denied environments [2]. In that work, the vision algorithms ran off-board on a ground station transferring the control commands wireless back to the helicopter. Since wireless communication is not always reliable, there is a strong need for computing all tasks of the control framework onboard.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 231855 (sFly). Markus Achtelik and Stephan Weiss are currently PhD students at the ETH Zurich (email: {markus.achtelik, stephan.weiss}@mavt.ethz.ch). Roland Siegwart is full professor at the ETH Zurich and head of the Autonomous Systems Lab (email: r.siegwart@ieee.org).

Michael Achtelik is CEO of Ascending Technologies GmbH, Germany (email: michael.achtelik@asctec.de)

Fig. 1. Outdoor autonomous flight using only onboard IMU aided monocular vision.

In this work, we develop our previous system to an onboard solution. As exteroceptive sensor, we still use one single camera because stereo vision loses its effect for large distances and small baselines. However, we include the onboard IMU more tightly as in the previous work, to tackle the issue of low-frequent vision updates. Also, we added a pressure sensor to estimate the absolute scale of the visual pose measurements. Finally, we optimized the visual framework for an embedded solution in order to cope with the limited calculation power onboard.

II. Related work

Previous work on position control using visual input has been done in several ways. Stable flights were shown using onboard cameras and landmarks placed in the environment such as blobs or other artificial markers in [3]. However, these approaches only work in controlled environments whereas we focus on stable flights without having any prior information about the environment nor GPS signals. This was successfully shown in [4] using a lightweight laserscanner and/or cameras in a stereo configuration and off-board computation. Offloading sensor data to a ground station has major drawbacks. Not only become delays a significant issue, but also the high bandwidth datalink has to be granted at any time. Compressing large data chunks (such as images) diminishes the issue but introduces artefacts. Furthermore, in the mentioned approach the stereo camera and the laser scanner have a limited range of operation.

An alternative approach is to use cameras for the localization task. However this vast information has to be processed accordingly. The most simple way is to install

a number of external cameras with known location and to have them track the MAV [5], [6], [7]. This method is very efficient for testing purposes and can be used to evaluate other approaches as ground truth reference. However it is not suitable for missions where the installation of an appropriate infrastructure is not feasible.

This approach can also be implemented the other way round: the camera is mounted on the helicopter and tracks a known pattern on the ground [8]. Hamel et al. [9] implemented a visual servoing based trajectory tracking to control an UAV with a camera observing n fixed points. Further methods have also been developed by fusing the visual data with IMU data [10].

Alternatively, stabilizing controllers can be built by means of optical flow considerations [11]. Herisse et al. [12] use an optical flow based PI-controller to stabilize a hovering MAV. They also implemented an automatic landing routine by contemplating the divergent optical flow. Hrabar et al. [13] developed a platform able to navigate through urban canyons. It was based on the analysis of the optical flow on both sides of the vehicle. Also, by having a forward looking stereo camera, they were able to avoid oncoming obstacles. Most recently, cheap systems sold as toys [14] were presented, performing stabilization based on optical flow and ultra sonic height sensors onboard.

An approach with offboard vehicle tracking equipment was implemented by Ahrens et al. [15]. Based on the visual SLAM algorithm of Davison et al. [16], they build a localization and mapping framework that is able to provide an almost drift-free pose estimation. With that they implemented a very efficient position controller and obstacle avoidance framework. However, due to the simplification they used in their feature tracking algorithm, a non-negligible drift persists. Also, they used an external Vicon localization system to control the aerial vehicle with millimeter precision (a system of external cameras that tracks the 3D pose of the vehicle). So far, they did not use the output of the visual SLAM based localization system for controlling the vehicle.

In this paper, we discuss the thorough implementation of a vision based controller framework onboard a micro helicopter. Compared to optical flow based approaches that drift over time, we focus on a solution that enables absolute position control. Note that we do not claim to have developed a novel controller. This has already been discussed in previous work [17], [7]. Rather, we highlight the issues of a full onboard implementation and their solutions. More precisely, our contributions are the following. First, we present a framework to tackle the issue of a very slow visual pose update versus the high agility of the micro helicopter. We solve this issue with filtering the visual information with inputs from inertial sensors at $1\,\mathrm{kHz}$ onboard the helicopter. This high frequency also enables us to estimate a reliable speed information that is crucial for such agile platforms. Since our visual framework consists of only one single fisheye camera, our second main contribution is to demonstrate how to recover the absolute scale of the visual pose estimation. We do so by filtering the visual pose with an onboard pressure sensor. The filter also compensates for the pressure sensor's drift. Last, we discuss a fast implementation of the visual framework [18] onboard the micro helicopter.

Our implementation of an onboard monocular vision-based MAV controller can be used in an unknown environment without the aid of any infrastructure based localization system, any beacons, artificial features, or any prior knowledge on the environment. In other words, our platform does not need any external assistance in order to navigate through an unexplored region and is not bound to a ground station. All our implementations are based on the Robot Operating System (ROS) [19]. This makes our work reusable for the community and represents as such a valuable contribution towards the fast development of autonomously flying MAVs.

The remainder of the paper is organized as follows: In Section III, we describe the platform we used. In Section IV, we present the algorithm and the implementation of the system. Experimental results and the evaluation are shown in Section V. Conclusions are given in Section VI.

III. Platform Description

A. Hardware

The MAV we use is a so-called quadrocopter, a helicopter driven by for rotors, symmetric to the center of mass. The control of the quadrocopter is performed solely by changing the rotation speed of the propellers and is described in more detail in [20]. For our experiments, we use the "AscTec Pelican" quadrocopter [1], which is a further development of the one described in [20]. The quadrocopter is equipped with rotors with 10" diameter which allow to carry a payload of about $500\,\mathrm{g}$. Depending on battery size and payload, flight times between 10 and 20 minutes can be achieved. Further key features are the Flight Control Unit (FCU) "AscTec Autopilot" as well as the flexible design enabling one to easily mount different payloads like computer boards or cameras. The FCU features a complete Inertial Measurement Unit (IMU) as well as two $32\,\mathrm{Bit}$, $60\,\mathrm{MHz}$ ARM-7 microcontrollers used for data fusion and flight control. One of these microcontrollers, the Low Level Processor (LLP) is responsible for the hardware management and IMU sensor data fusion. An attitude and GPS-based position controller is implemented as well on this processor. The LLP is delivered as a black box with defined interfaces to additional components and to the High Level Processor (HLP). To operate the quadrocopter, only the LLP is necessary. Therefore, the HLP is dedicated for custom code. All relevant and fused IMU data is provided at an update rate of $1\,\mathrm{kHz}$ via a highspeed serial interface. In particular, this comprises body accelerations, body angular velocities, magnetic compass, height measured by an air pressure sensor and the estimated attitude of the vehicle.

For the computationally more expensive onboard processing tasks, we outfitted the helicopter with a 1.6 GHz Intel Atom Based embedded computer, available from [1]. This computer is equipped with 1 GB RAM, a MicroSD card slot for the operating system, a 802.11n based miniPCI Express WiFi card and a Compact Flash slot. The miniPCIE WiFi

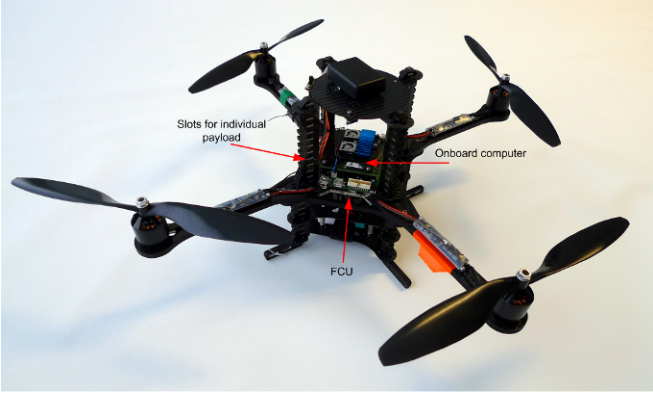

Fig. 2. Overview of the Pelican quadrocopter

card is preferred over USB to keep the USB bus free for devices like the cameras we use. We furthermore use a high speed CF-card that allows us data logging with up to 40 MByte/s.

As camera, we use a Point-Grey USB Firefly camera with a resolution of 752×480 px and a global shutter. The camera faces the ground with a $150°$ field-of-view lens since we are expecting the most stable features trackable over longer time in this configuration.

In this work, a position controller and the position data fusion algorithms are implemented on the HLP, based on the vision input from the onboard computer and the inertial data provided by the LLP. On the LLP, the attitude controller is used as inner loop. A GPS-based position controller is used as a fall-back in case of any failures on the HLP during outdoor experiments.

B. Software

To provide a maximum portability of our code and to avoid potential (binary) driver issues, we installed Ubuntu Linux 10.04 on our onboard computer which makes tedious crosscompiling unnecessary. Since we are running a couple of different subsystems that need to communicate between each other, we use the ROS [19] framework as a middleware. This is also used to communicate to the ground station over the WiFi datalink for monitoring and control purposes. The FCU is interfaced via a ROS node communicating over a serial link to the FCU's Higlevel Controller with firmware we developed for our purposes.

Software development on the HLP is done based on a SDK available for the AutoPilot FCU providing all communication routines to the LLP and a basic framework. The HLP communicates with the ROS framework on the onboard computer over a serial datalink and a ROS FCU-node handling the serial communication. This node subscribes to generic ROS pose messages with covariance, in our case from the vision framework, and forwards it to the HLP. Moreover, it allows to monitor the state of the fusion filter and the position controller, and to adjust their parameters online via the "dynamic reconfigure" functionality of ROS.

For the implementation of the position control loop and data fusion onboard the HLP, a Matlab/Simulink framework is used in combination with the Mathworks Real-Time Workshop Embedded Coder. The framework provides all necessary tools to design the control structure in Simulink, optimize it for fixed point computing, as well as compiling and flashing the HLP.

IV. Onboard Vision Based Position Control

A. Overview

In this section, we describe the essential components that we used to enable autonomous flights running all computation onboard. The basic structure can be seen in Fig. 3. We obtain absolute position estimates by a monocular visual SLAM (VSLAM) framework and estimate the absolute scale with the help of an air pressure sensor. Since this process (approx. $10\,\mathrm{Hz}$) is slow compared to the motion of the MAV, we fuse this information with inertial sensor data (angular rates and body acceleration) provided by the IMU at a rate of $1\,\mathrm{kHz}$. The outputs of that filter are finally fed into a position controller based on nonlinear dynamic inversion. While the computationally expensive VLSAM is run on the Atom onboard computer at approximately $10\,\mathrm{Hz}$, the fusion filter and the position controller are executed on the HLP (see Section III-A) at $1\,\mathrm{kHz}$ – just when new IMU readings arrive. This ensures minimum possible delays and allows us to handle the fast movements and disturbances of the MAV. The ground station is solely used for monitoring or sending highlevel commands such as waypoints.

B. Visual framework

The approach presented in this paper uses the visual SLAM (VSLAM) algorithm of Klein and Murray [18] in order to localize the MAV with the aid of a single camera

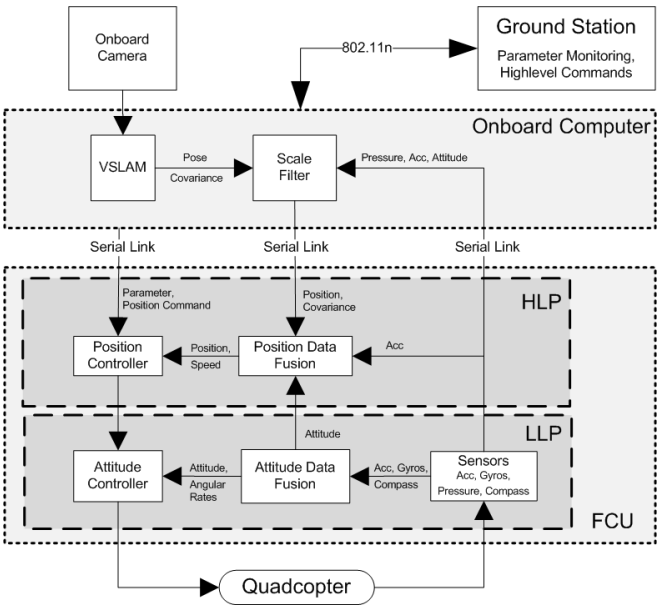

Fig. 3. System overview

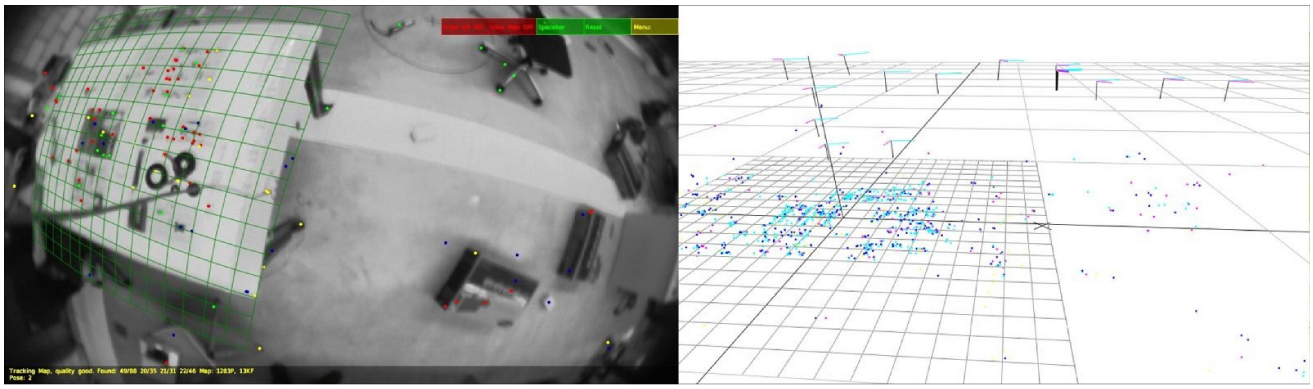

Fig. 4. Screenshot of Klein and Murray's SLAM algorithm. On the left, the tracking of the FAST corners can be observed, this is used for the localization of the camera. On the right, the 3D map that was build by the mapping thread is shown. The 3-axis coordinate frames represent the location where new keyframes where added.

(see Fig. 4). In short, the authors split the simultaneous localization and mapping task into two separately-scheduled threads: the tracking thread and the mapping thread. Splitting the SLAM algorithm into a mapping and a tracking part brings the advantage that both can run at different speeds. The tracker can thus yield fast pose updates while the mapper can use more powerful (slower) map optimization techniques. Compared to frame-by-frame SLAM the mapper does not process every camera frame. This eliminates to a great extend redundant information processing during slow movements or hovering. Furthermore, it is very easy to adapt and optimize independently each of the threads to our specific needs on the flying platform. These are the main reasons we choose this SLAM algorithm. We describe our modifications in the following.

As our whole framework uses ROS as middleware, we modified the VSLAM such that it exports the 6-DOF pose estimate, the map information and tracking quality as ROS messages to the HLP over the FCU-node. Similarly, we modified it to accept initialization and reset commands as ROS services such that we can remote control the algorithm from the ground station. Note that only during the VSLAM initialization phase these commands are sent from the ground station to the MAV. The VSLAM framework itself runs completely on the onboard computer.

The tracking part of the VSLAM algorithm is already well designed for our needs. We set it to use a maximum of 300 features per frame. For successful MAV navigation we only need our vincinity to be consistent, that is, it is sufficient to have only a local map well aligned with the gravity vector. The FCU's gravity estimate and the air pressure sensor can be used to compensate for map and scale drifts respectively (see Section IV-C). More important is that we do not have to store a global map. In fact only a few keyframes are sufficient for a local consistent pose estimate. Here, we limit the number of keyframes retained in the map. When a new keyframe is requested, the keyframe furthest away in the euclidean space is deleted. Also, all features corresponding to that keyframe are removed from the map. This ensures constant speed in map maintenance since only N keyframes and M features take part in the nonlinear map refinement. Also, it ensures constant speed in tracking in already explored areas since the number of features M is roughly constant.

With these modifications, the VSLAM algorithm runs approximately at $10\,\mathrm{Hz}$ on the onboard computer. The framerate can temporarily drop down to $5\,\mathrm{Hz}$ during a nonlinear map refinement when a new keyframe is added. Of course a loss of the local map or textureless regions are fatal for the here presented algorithm. However, failure of the VSLAM algorithm can be detected. In that case, the data fusion algorithm from Section IV-D is not updated any more. The vehicle will then drift away slowly since position information just relies on integration of acceleration sensors. This still leaves enough time for a safety pilot to take over control or for switching back to an alternative localization. Thus far, we experienced that in outdoor environments we rarely lack of features. Known difficult scenarios are environments like self-similar paved roads or uniform indoor floors, while natural scenes, such as bushes or unpaved roads usually provide sufficient texture.

C. Scale Estimation

Observe that, because we are using a single camera, the VSLAM framework can give us only the direction of translation but not its magnitude, that is, the absolute scale.

To recover the absolute scale—in order to pass proper position information to the fusion filter—there are basically two solutions. The first solution consists in measuring the size of an element in the scene. This quickly gets computationally infeasible on our setup and is very likely prone to errors. The other solution is to use additional sensors that provide absolute measurements. For instance, previous work has been done with ultrasonic range sensors. This works well in principle but limits the maximum operating height of the vehicle which is about $2-5\,\mathrm{m}$ for commonly available sensors.

Therefore, we use the accelerometers and the air pressure sensor of the FCU to recover the scale. The pressure sensor has the advantage of almost unlimited height, but the drawbacks are drift and noisy measurements. Simply using the

ratio of the height measured by the air pressure sensor and the (also noisy) height from VSLAM would lead to inaccurate results. Potential issues with taking measurements from the air pressure sensor are twofold. First, the zero height of the pressure sensor does not align with the VSLAM reference frame. Second, the measurements of the pressure sensor are drifting over time due to changing weather conditions. We solve these problems by designing an EKF using the pressure sensor as well as the accelerometers, and incorporating the scale and pressure sensor drift in the states.

The state $\boldsymbol{x}$ consists of the absolute height p_z, the climb rate v_z, the absolute scale λ and the pressure sensor bias b. As process input, we chose the acceleration a_z expressed in world coordinates. To gain the acceleration $\boldsymbol{a}$ in world coordinates from the measured body acceleration $\boldsymbol{a}_{body}$, we need to transform it by the attitude $\boldsymbol{R} \in SO(3)$ of the vehicle, estimated by the FCU. Finally, $\boldsymbol{a}$ needs to be corrected for the gravity.

$$\boldsymbol{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T = \boldsymbol{R} \cdot \boldsymbol{a}_{body} - \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T \quad (1)$$

As measurement $\boldsymbol{z}$, we chose the height from VSLAM $p_{z,v}$ and the height measured by the air pressure sensor $p_{z,p}$. To summarize:

$$\boldsymbol{x} = \begin{bmatrix} p_z & v_z & \lambda & b \end{bmatrix}^T \quad \boldsymbol{z} = \begin{bmatrix} p_{z,p} & p_{z,v} \end{bmatrix}^T \quad (2)$$

The differential equations governing the state are:

$$\dot{p}_z = v_z \qquad \dot{b} = n_b \quad (3)$$

$$\dot{v}_z = a_z + n_a \qquad \dot{\lambda} = n_\lambda \quad (4)$$

The noise n_a of the acceleration measurement is assumed to be white gaussian noise. Bias b and scale λ are modeled as random walks with their derivatives being white gaussian noise n_a and n_λ respectively. We have two measurements arriving not synchronized and at different rates, therefore we need measurement prediction functions $h_v(\boldsymbol{x})$ and $h_p(\boldsymbol{x})$ for the scaled height measurement $p_{z,v}$ from the vision algorithm and the absolute height measurement $p_{z,p}$ from the pressure sensor:

$$\hat{z}_v = h_v(\boldsymbol{x}) = p_z \cdot \lambda; \qquad \tilde{z}_v = p_{z,v} - h_v(\boldsymbol{x}) \quad (5)$$

$$\hat{z}_p = h_p(\boldsymbol{x}) = p_z + b; \qquad \tilde{z}_p = p_{z,p} - h_p(\boldsymbol{x}) \quad (6)$$

State update, Kalman gain and process covariance are finally computed following the standard EKF scheme. To estimate the bias properly, motion of the vehicle in the z axis is required, otherwise the scale will not be correctly estimated. The performance of the filter will be evaluated in Section V

D. Data Fusion

Fast data fusion algorithms are essential to match the high bandwidth of the quadrocopter's system dynamics. The attitude angles and angular rates of the quadrocopter are already provided by the LLP at $1\,\mathrm{kHz}$ update rate. For position control, fast data fusion algorithms of all kinematic measurements are needed. The LLP provides acceleration measurements at $1\,\mathrm{kHz}$ update rate and the vision system provides position and heading information at $5 - 10\,\mathrm{Hz}$. A position filter has been developed taking into account the computational limitations of the microcontroller hardware where a full state Kalman filter working at an update rate of at least $500\,\mathrm{Hz}$ is not feasible. This high update rate is needed to enable a high update rate in the position control loop to match the quadrocopter's system dynamics. In particular, the aim of the filter is to combine both the vision and the acceleration sensor to achieve a fused signal, featuring fast reactions on disturbances based on the high update rate of the acceleration sensors and steady state accuracy based on the vision signal. The filter is designed decoupled for all three axes x, y, z and works in a global (0-) frame. In the following, only the filter for the x-axis is described and applies for the other axes respectively. The body-fixed accelerations are rotated in the global frame by a simple rotational matrix based on the attitude angles provided by the LLP, as shown in (1). The filter is based on a Luenberger observer [21] with the position p_x, speed v_x, and the acceleration sensor bias b_x as state. The acceleration notated in the global (0-) frame, separated for each axis, is the system input. The measurement can be any (absolute) position input, which in our case is the position $\boldsymbol{p}_{v\lambda} = \boldsymbol{p}_v/\lambda$ from the VSLAM, corrected by the scale estimated with the method described in Section IV-C.

$$\boldsymbol{x} = \begin{bmatrix} p_x & v_x & b_x \end{bmatrix}^T \quad \boldsymbol{u} = \begin{bmatrix} a_x \end{bmatrix}^T \quad \boldsymbol{y} = \begin{bmatrix} p_{x,v\lambda} \end{bmatrix}^T \quad (7)$$

Again, we use a linear motion model to describe the system. Since this filter runs at 1 kHz, we consider this as continuous time system.

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A} \cdot \hat{\boldsymbol{x}} + \boldsymbol{L}\,(\boldsymbol{y} - \hat{\boldsymbol{y}}) + \boldsymbol{B} \cdot \boldsymbol{u} \quad (8)$$

$$\hat{\boldsymbol{y}} = \boldsymbol{H} \cdot \hat{\boldsymbol{x}} \quad (9)$$

with:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad \boldsymbol{L} = \begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix}^T$$

The elements of the matrix L are calculated based on considerations on the eigenvalues of the error dynamics. The state error is defined as $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \hat{\boldsymbol{x}}$ and the error dynamics can be calculated as:

$$\dot{\tilde{\boldsymbol{x}}} = (\boldsymbol{A} - \boldsymbol{L}\boldsymbol{H})\,\tilde{\boldsymbol{x}} \quad (10)$$

We used Simulink and the Mathworks Real-Time Workshop Embedded Coder to implement the data fusion filter on the HLP (see Section III-B). All calculations are optimized for fixed point arithmetics and unnecessary matrix operations are dropped. Relevant parameters such as $\boldsymbol{L}$ as well as the states are connected to communication channels that can be accessed through ROS messages and services (see Section III-B) for debugging and parameter changes. In future, it is planned to use this functionality to extend the observer to a full state Kalman filter by computing the update step on the onboard computer to find optimal values for $\boldsymbol{L}$ when a new measurement arrives.

In a last step, methods were implemented to reset, hold and reinitialize the filter's integrators in case of loss or re-initialization of the input from VSLAM. The output of this position filter are position and speed signals. The filter is able to react fast to disturbances measured by the acceleration sensors far before it is possible to observe these disturbances by the visual sensor.

E. Position Controller

For position control, a cascade structure is used. As inner loop, the well tested attitude loop provided by the LLP of the FCU is used (see Section III-A). The outer loop is the position loop, and is implemented on the HLP based on the concept of nonlinear dynamic inversion. With an adequate knowledge of the plant dynamics, this control approach can transform the nonlinear system into an equivalent linear system without any simplification, through exact state transformation and suitable control inputs [22]. Based on this input-output linearization, standard linear control strategies like PD controllers can be applied. For the quadrocopter position controller, a control structure of relative degree two is implemented. That means, position and speed control are performed in one control loop on the onboard hardware. Fig. 5 shows the control structure including the rates of the different parts.

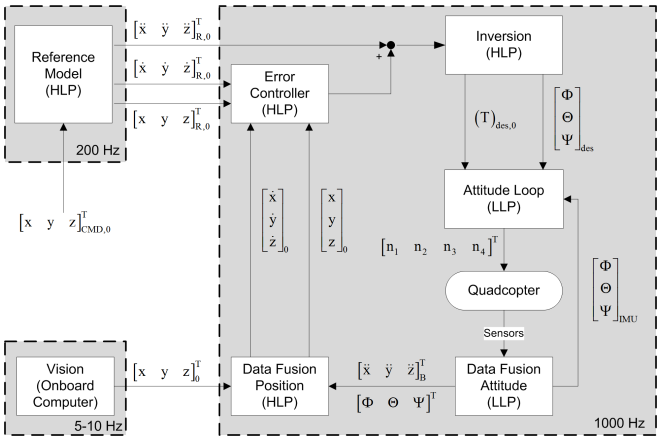

Fig. 5. Structure of the position controller. Subscript 0 denotes coordinates w.r.t a global frame, B w.r.t the current body frame. The names in braces denote on which physical device the corresponding part is executed

Input commands of the attitude loop are desired attitude angles roll, pitch and yaw $\begin{bmatrix}\Phi & \Theta & \Psi\end{bmatrix}^T_{des}$ and the thrust T_{des} commanded by the position loop. Outputs are the commanded rotational velocities $\begin{bmatrix}n_1 & \dots & n_4\end{bmatrix}^T$ of the four rotors. This control loop is implemented on the LLP, delivered with the FCU and is not the focus of this paper.

For the position control loop, the quadrocopter translation dynamics need to be modeled and inverted. The world frame (denoted by 0) is used as inertial frame in order to apply to Newton's law. Furthermore, the data fusion and the generation of reference trajectories, as described later, is performed in this frame. To simplify the inversion, the $\bar{0}$-frame has been introduced as a leveled frame with the same yaw angle Ψ as a local body frame denoted by B. In a first step, desired accelerations in the 0-frame can be transformed into the $\bar{0}$-frame by a simple rotation through the azimuth Ψ. Applying Newton's second law results in:

$$m \cdot \boldsymbol{a}_{\bar{0}} = \boldsymbol{f}_{\bar{0}} + \boldsymbol{f}_{g,\bar{0}} = \boldsymbol{M}_{\bar{0}B} \cdot \boldsymbol{f}_B + \boldsymbol{f}_{g,\bar{0}} \tag{11}$$

m denotes the mass of the quadrocopter, $\boldsymbol{a}$ the acceleration expressed in the $\bar{0}$-frame, $\boldsymbol{f}$ the forces on the quadrocopter in the $\bar{0}$-frame and the B-frame respectively, $\boldsymbol{f}_g$ the gravitational vector and $\boldsymbol{M}_{\bar{0}B}$ denotes the transformation matrix between the $\bar{0}$ and B-frame. Solved for the roll angle Φ, the pitch angle Θ and the thrust T, we get:

$$T = m \cdot \sqrt{a_x^2 + a_y^2 + (a_z - g)^2} \tag{12}$$

$$\Phi = \arctan \frac{m a_y}{T} \tag{13}$$

$$\Theta = \arctan \frac{a_x}{a_z - w} \tag{14}$$

Where g is the gravitational constant. By these equations, a transformation is given transforming so called pseudo controls $\boldsymbol{\nu} = \boldsymbol{a}$ into the controls of the system Φ, Θ, T. Therefore, a linear dynamic between the inputs, the position commands and the pseudo controls is achieved and linear control methods can be applied.

As only the second time derivatives of the position commands $\boldsymbol{p}_c$ can be commanded as pseudo controls, the command trajectory needs to be smooth such that its second time derivative exists. Therefore, linear reference models are used to generate the reference trajectories $\boldsymbol{p}_R$, computed in the 0-frame and governed by the following equation:

$$\ddot{\boldsymbol{p}}_R = \omega_0^2 \cdot (\boldsymbol{p}_c - \boldsymbol{p}_R) - 2\zeta\omega_0 \cdot \dot{\boldsymbol{p}}_R \tag{15}$$

The reference dynamics can be set by the natural frequency ω_0 and the relative damping ζ. For the controller presented in this paper, the damping is set to 1 to ensure aperiodic behavior and the natural frequency to 2.5 based on experiments. The error controller, computing the pseudo controls can now be designed and is governed by the following equation:

$$\boldsymbol{\nu} = \ddot{\boldsymbol{p}}_R + (\dot{\boldsymbol{p}}_R - \dot{\boldsymbol{p}}) \cdot k_d + (\boldsymbol{p}_R - \boldsymbol{p}) \cdot k_p \tag{16}$$

Where $\boldsymbol{p}$ is the filtered position and $\dot{\boldsymbol{p}}$ the filtered velocity from Section IV-D. k_p, k_d are the proportional and differential gains for the error controller.

At this point, one advantage of the system becomes obvious: Commanding a step signal for the position, a trajectory for acceleration, speed and position is computed. The trajectory for the acceleration is directly commanded to the vehicle and the trajectories for speed and position are controlled by the error controller. This results in acceleration at the beginning with maximum allowed value and slowing down with exactly the maximum available acceleration as well. An example of the generated reference trajectories is shown in Fig. 6

For the implementation, the Simulink framework is used as well, and all calculations are optimized for fixed-point arithmetics. Furthermore, limitations are introduced to limit

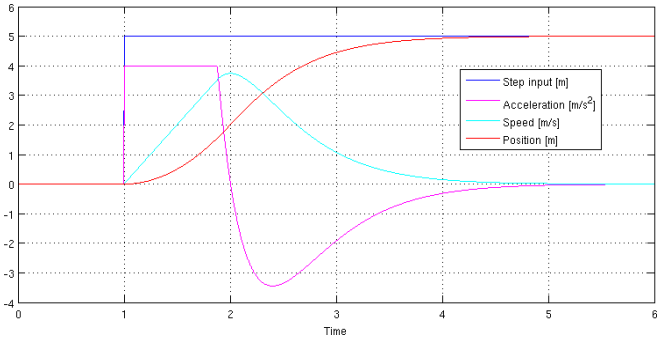

Fig. 6. Response of the reference model on an input step, in our case a change in the desired position. Note that the model outputs a negative acceleration after $t = 2\,\mathrm{s}$ to slow the vehicle down in advance in order to arrive the desired position fast and without overshoot. The acceleration was limited to $4\,\mathrm{m/s^2}$

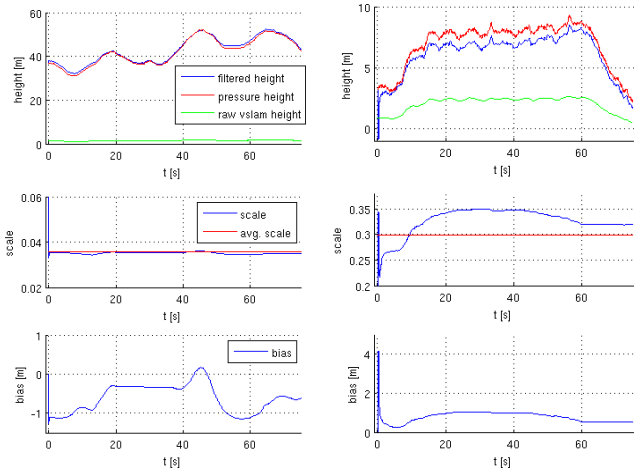

Fig. 7. Results of the scale filter for different heights. Top: estimated height from the filter (blue), raw height from VSLAM (green) and the raw height from the pressure sensor for comparison (red). Middle: absolute estimated scale compared to the average scale of the whole flight. Bottom: estimated sensor bias. Note the different scales on the plots

accelerations and attitude angles to reasonable values. A command filter was also implemented, giving the MAV's safety pilot the possibility to steer speed signals in the $\bar{0}$-frame with the sticks of his RC transmitter.

V. Experiments and Results

In this section, we evaluate the performance of the presented system, in particular of the scale filter and the position controller. The indoor experiments were made in the "Flying Machine Arena" [23] equipped with a Vicon motion capturing system which provided us ground truth with millimeter resolution at $200\,\mathrm{Hz}$. The outdoor experiments took place in a small park with grass and stones on the ground which provided sufficient texture.

The top plots of Fig. 7 show the height estimated by our filter (blue) compared to the raw measurements of the air pressure sensor (red) and the raw estimated height from the VSLAM in "VSLAM-units" (green). The middle plots show the estimated scale compared to the ratio of the mean height from the air pressure sensor and from the VSLAM. The estimated bias to compensate for the drifts of the air pressure sensor can be seen in the bottom plots. We initialized the filter with constant parameters at different height to verify that it still converges under different conditions. We unfortunately did not have ground truth accurate enough for these experiments. Therefore, we can only evaluate the plots qualitatively. What can be observed is that the filter converges after approximately $1\,\mathrm{s}$. For the pressure sensor bias, comparable drifts are observable when the vehicle is left on the ground over the same period of time.

For the position controller, the RMS error while hovering and the response to external disturbances or step inputs respectively is of interest. Fig. 8 and Fig. 9 show the trajectories of in- and outdoor flights. To gain the RMS error for the outdoor experiments, we computed the mean of the hovering phases and computed the error relative to this mean. Obviously, the RMS error outdoor is larger than indoor which is a result of the higher altitude and the less controlled environment, but the vehicle is still able to navigate stably. It can also be observed that the RMS error in the z-axis is significantly smaller than in the x/y-plane as depicted in Table I. This is because of the controller structure and dynamics of the MAV. While we can directly command acceleration in the z-axis through the thrust of the propellers, commands in the x/y-plane need to go through the attitude control loop first, and then result in acceleration (see Section IV-E, Fig. 5).

TABLE I
RMS error while hovering

Type	RMS error [m]	Height [m]
indoor x/y	0.069	1.4
indoor z	0.009	1.4
outdoor x/y	0.44	3.3
outdoor z	0.11	3.3

The left plot of Fig. 10 shows disturbances at $t = 6\,\mathrm{s}$ and $t = 20\,\mathrm{s}$ in the y-axis from pushing the vehicle. Note that the vehicle directly returns to its desired setpoint within the RMS with almost no overshoot. This is a result from the fast data fusion and of the position controller based on nonlinear dynamic inversion (see also Fig. 6). The middle

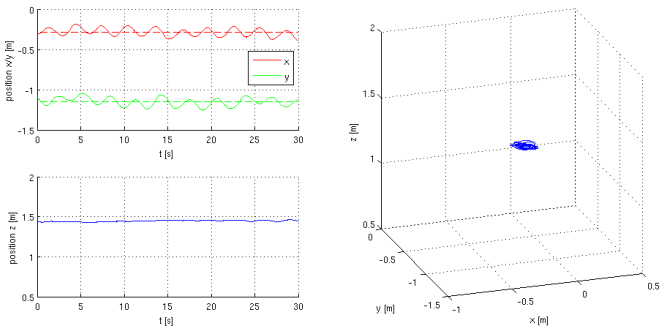

Fig. 8. Indoor hovering

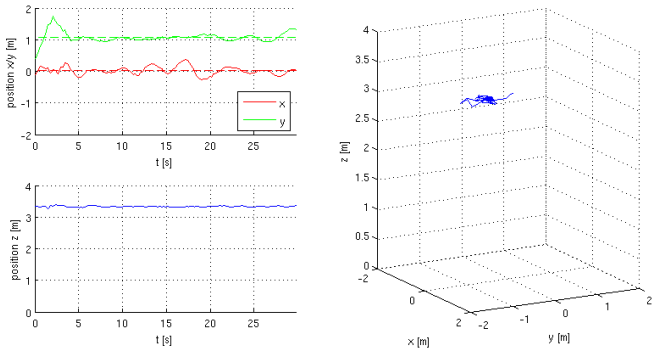

Fig. 9. Outdoor hovering

plot of Fig. 10 shows a longer term disturbance at $t = 14\,\mathrm{s}$ by pulling the vehicle and holding it for $3\,\mathrm{s}$. Again, the vehicle returns directly to the desired setpoint with almost no overshoot. The disturbances applied in the z-axis on the right plot of Fig. 10 look rather small. Because of the direct thrust command as explained above, the helicopter was massively working against the disturbance. Even though we disturbed the vehicle with reasonably strong impulses, we did not manage to deflect it more than $20\,\mathrm{cm}$.

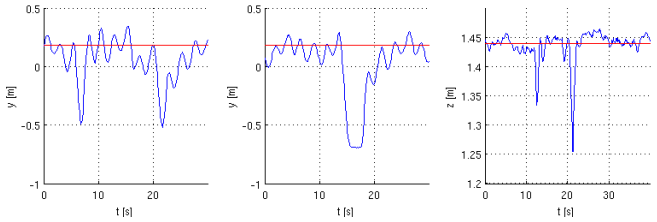

Fig. 10. Short and long disturbances in the y-axis on the left and the middle. Disturbances in the z-axis on the right. Note the different scale on the right plot

VI. Conclusions

We successfully stabilized a highly dynamic aerial vehicle based on onboard vision computation at a rate of only $10\,\mathrm{Hz}$, data fusion with IMU data and a well designed implementation. With the frameworks we developed, we are now able to try out relatively easy new control approaches on a real working system. This system can perform autonomous flights in unknown in- and outdoor environments, solely having a monocular camera as exteroceptive sensor while all computation is completely running onboard.

VII. Acknowledgments

We like to thank Prof. Raffaello D'Andrea and his group for giving us access and support for their "Flying Machine Arena" [23] to obtain ground truth for the indoor experiments. We also like to thank Prof. Florian Holzapfel of the Institute of Flight System Dynamics at the Technische Universität München and The Mathworks for their great cooperation concerning the used Simulink Framework for the microprocessor of the onboard FCU and the nonlinear control structures for the quadrocopter.

References

[1] Ascending Technologies GmbH, website, http://www.asctec.de.
[2] M. Bloesch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *IEEE International Conference on Robotics and Automation*, 2010.
[3] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision based position control for mavs using one single artificial landmark," in *International Conference & Exhibition on Unmanned Aerial Vehicles (UAV)*, 2010.
[4] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments," in *Proceedings of the SPIE Unmanned Systems Technology XI*, 2009.
[5] E. Altug, J. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *International Conference on Robotics and Automation*, May 2002, pp. 72–77.
[6] S. Park, D. Won, M. Kang, T. Kim, H. Lee, and S. Kwon, "Ric (robust internal-loop compensator) based flight control of a quad-rotor type uav," in *International Conference on Intelligent Robots and Systems*, Aug. 2005, pp. 3542–3547.
[7] S. Klose, J. Wang, M. Achtelik, G. Panin, F. Holzapfel, and A. Knoll, "Markerless, Vision-Assisted Flight Control of a Quadrocopter," in *International Conference on Intelligent Robots and Systems*, October 2010.
[8] B. Ludington, E. Johnson, and G. Vachtsevanos, "Augmenting uav autonomy," *IEEE Robot. Automat. Mag.*, vol. 24, no. 5, pp. 63–71, Sept. 2006.
[9] T. Hamel, R. Mahony, and A. Chriette, "Visual servo trajectory tracking for a four rotor vtol aerial vehicle," in *International Conference on Robotics and Automation*, May 2002, pp. 2781–2786.
[10] T. Cheviron, T. Hamel, R. Mahony, and G. Baldwin, "Robust nonlinear fusion of inertial and visual data for position, velocity and attitude estimation of uav," in *International Conference on Robotics and Automation*, Apr. 2007, pp. 2010–2016.
[11] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "Mav obstacle avoidance using optical flow," in *International Conference on Robotics and Automation*, May 2010.
[12] B. herisse, F. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow," in *International Conference on Intelligent Robots and Systems*, Sept. 2008, pp. 801–806.
[13] S. Hrabar, G. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a uav," in *International Conference on Intelligent Robots and Systems*, Aug. 2005, pp. 3309–3316.
[14] Parrot S.A., "AR.Drone," website, http://ardrone.parrot.com.
[15] S. Ahrens, D. Levine, G. Andrews, and J. How, "Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments," in *International Conference on Robotics and Automation*, May 2009.
[16] A. Davison, I. Reid, N. Molton, and O. Strasse, "Monoslam: Real-time single camera slam," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 6, pp. 1052–1067, June 2007.
[17] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *International Conference on Robotics and Automation*, Apr. 2005, pp. 2247–2252.
[18] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *International Symposium on Mixed and Augmented Reality*, Nov. 2007, pp. 225–234.
[19] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.
[20] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 khz," in *IEEE International Conference on Robotics and Automation*, Roma, Italy, Apr. 2007, pp. 361 – 366.
[21] K. Narendra and A. Annaswamy, *Stable Adaptive Control*. Prentice Hall, 1990.
[22] A. Isidori, *Nonlinear Control Systems*, 3rd ed. Springer, 1995.
[23] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *IEEE International Conference on Robotics and Automation*, May. 2010.