On-board Velocity Estimation and Closed-loop Control of a Quadrotor UAV based on Optical Flow

Volker Grabe, Heinrich H. Bülthoff, and Paolo Robuffo Giordano

Abstract— **Robot vision became a field of increasing importance in micro aerial vehicle robotics with the availability of small and light hardware. While most approaches rely on external ground stations because of the need of high computational power, we will present a full autonomous setup using only on-board hardware. Our work is based on the *continuous homography constraint* to recover ego-motion from optical flow. Thus we are able to provide an efficient fall back routine for any kind of UAV (Unmanned Aerial Vehicles) since we rely solely on a monocular camera and on on-board computation. In particular, we devised two variants of the classical continuous 4-point algorithm and provided an extensive experimental evaluation against a known ground truth. The results show that our approach is able to recover the ego-motion of a flying UAV in realistic conditions and by only relying on the limited on-board computational power. Furthermore, we exploited the velocity estimation for closing the loop and controlling the motion of the UAV online.**

I. INTRODUCTION

In the recent years, vertical take-off and landing vehicles became a very popular focus of research among roboticists. Especially quadrotors combine high agility with a reasonable amount of payload. Thus, contrarily to autonomous fixed wing aircrafts, they are able to navigate in small cluttered spaces. This allows the use of quadrotors in indoor environments or unaccessible/hostile locations such as sites affected by natural catastrophes.

In order to provide sufficiently reliable spatial localization in these scenarios, robotic vision as intersection between robotics and computer vision is currently an active field of research. Cameras can be very light weighted and provide a rich sensory feedback in a large range of environments and conditions. Therefore, they are particularly suitable for use on a flying vehicle. Additionally, the information perceived by a camera can be used for many different applications starting from leader following and automatic landing [1], [2], up to a full three dimensional mapping of the environment [3].

However, many recent advanced results from the field of computer vision were achieved thanks to the availability of high computational power and, often, not meeting the hard real-time performance required for closed-loop control of robotic systems [3]. With the aim of developing vision systems which are suitable for real-time control, several projects inspired by the development of augmented reality approaches were recently presented [4], [5], [6]. However, all of these visual SLAM (simultaneous localization and mapping) setups rely on the possibility to forward demanding large portions (if not all) of the needed computations to an ad-hoc ground station. This, however, greatly reduces the flexibility of the robotic system at hand. Additionally, they usually do not include a reliable emergency backup behavior in case of lost tracking, an unwanted but common situation when dealing with artificial vision.

Up to now, only a few real-time approaches are able to cope with the limited processing power available on current on-board hardware. However, one of the first system with all processing done on-board uses a laser scanner as main data source to detect the environment [7]. A camera is used only with a frequency of 2 Hz to detect loop closures. Unfortunately, compared to cameras, laser scanners can only observe a two dimensional slice of the world and are much more demanding for on-board use in terms of weight and energy consumption. To the best of our knowledge, only one solution has been presented so far combining the use of a camera with pure on-board processing [8]. The authors exploit a reduced version of PTAM to create a sparse local map. However, there is still no backup behavior to deal with tracking failures.

Other approaches try to detect features of known position in the environment in order to recover a self-position estimate [9], [10]. While this setup could be moved between different locations, it is still limited by the range in which features are located. Furthermore, these systems require a certain number of known features to be visually detected at any time in order to maintain a good tracking performance.

In our work, we will address several of the aforementioned issues by relying on direct self-motion estimation from optical flow. While optical flow was used before on UAVs to allow velocity estimation without the need to maintain a map, this possibility was seldom demonstrated in actual operation of real robots [11], in closed loop control [12], using only on-board hardware [13]. A notable exception is a recent commercial system, the Parrot[1] AR.Drone, which makes use of optical velocity estimation together with a sonar to stabilize itself in the environment. Along these lines, we present a real-time, low-computational method which can be used to hover a UAV in emergency situations and does neither rely on a map nor some known features in the image.

The rest of the paper is structured as follows: in Sect. II, we will describe the theoretical foundations of our approach

V. Grabe, and P. Robuffo Giordano are with the Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany {volker.grabe, prg}@tuebingen.mpg.de.

H. H. Bülthoff is with the Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany, and with the Department of Brain and Cognitive Engineering, Korea University, Seoul, 136-713 Korea. E-mail: hhb@tuebingen.mpg.de.

[1] www.parrot.com

and illustrate the main feature of our solution. Afterwards, in Sect. III, we will present our platform and report the results of the conducted experiments, which will then be discussed in Sect. IV. Section V will draw the concluding remarks and line out our future work.

II. SELF-MOTION ESTIMATION FROM OPTICAL FLOW

Estimation of self-(camera-)motion form optical flow is a longstanding problem which has been extensively addressed in the robotics and computer vision communities [14].

As opposed to more computationally expensive solutions such as visual SLAM, whose broad aim is to recover the 3D structure of an observed local environment, use of optical flow allows to obtain a direct (instantaneous) estimation of the camera translation/rotation among consecutive frames. This information is usually both exploited in more sophisticated reconstructions such as the aforementioned SLAM problem, but also as a direct feedback of the camera displacement (velocity) w.r.t. the surrounding environment. Indeed, many works have exploited this information in building simple/reactive obstacle avoidance algorithms or self-motion/velocity stabilization laws, see, e.g., [11].

In this paper, we focus on this latter case and propose an experimental validation of a self-motion estimation algorithm based on optical flow extraction. Experiments are carried on a typical UAV platform, a quadcopter carrying onboard a camera and an Inertial Measurement Unit (IMU) sensor. A relevant feature of our work, that distinguishes it from most of the past literature, is that we build upon the framework of *continuous motion recovery*, in particular of the so-called *continuous homography constraint* [15]. In fact, an often overlooked problem in self-motion estimation from optical flow is that one assumes small but *finite* camera displacements over frames. This gives rise to the well-known reconstruction methods based on the (discrete) epipolar/homography constraints whose aim is to recover a finite relative camera translation/rotation. However, since most cameras acquire images at high rates ($30 - 60\,\mathrm{Hz}$), in most robotics applications one is more interested in the camera instantaneous linear/angular velocity rather than in a finite displacement over time. Therefore, it seems more appropriate to adopt a *continuous* estimation point of view when trying to recover the camera self-motion.

In our experimental setup, we assume a downfacing camera in an approximately flat outdoor or indoor scenario. Features on the ground will be tracked between any two consecutive frames to compute an optical flow field $\Phi = ((\boldsymbol{x}_1, \boldsymbol{u}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{u}_n))$ as function of n pairs $(\boldsymbol{x}_i, \boldsymbol{u}_i)$ of detected features on the image plane $\boldsymbol{x}_i \in \mathbb{R}^3$ and associated image velocities $\boldsymbol{u}_i \in \mathbb{R}^3$. In indoor hallways as well as in many outdoor scenarios in open space, one can safely assume that any tracked feature will be located on the horizontal ground plane when using a downfacing camera. This assumption will be exploited as follows: after reviewing an algorithm to retrieve the linear and angular velocities ($\boldsymbol{v} \in \mathbb{R}^3, \boldsymbol{\omega} \in \mathbb{R}^3$) of a moving camera (w.r.t. the world frame and expressed in the camera frame) based on the classical continuous four-point algorithm for planar scenes [15], we will show how to extend it in the cases of (i) known angular velocities from onboard gyroscopes, and (ii) additional known direction of the ground plane normal vector in the camera frame. This is motivated by the fact that, apart from a measurement of $\boldsymbol{\omega}$, typical IMUs are also able to sense the local direction of gravity which, given our assumptions, coincides with the ground plane normal vector.

We will then present an experimental validation and thorough comparison of these three methods against a known ground truth. Finally, we will show the experimental results of using our proposed solutions as a feedback term for closed loop control of a quadrotor UAV.

As a first step, we now review the classical reconstruction algorithm based on the continuous homography constraint [15].

A. Review of the continuous homography constraint

The apparent velocity of a point $\boldsymbol{X} \in \mathbb{R}^3$ in space because of camera motion, and seen from camera frame, is

$$\dot{\boldsymbol{X}} = \hat{\boldsymbol{\omega}}\boldsymbol{X} + \boldsymbol{v} \tag{1}$$

where $\hat{\boldsymbol{\omega}} \in so(3)$ is the skew-symmetric matrix associated to the vector $\boldsymbol{\omega} \in \mathbb{R}^3$.

As discussed above, we assume that all features are located on a plane of equation $\boldsymbol{N}^T\boldsymbol{X} = d$ where $\boldsymbol{N} \in \mathbb{S}^2$ is the unit normal vector to the plane, and $d \in \mathbb{R}$ the plane distance from the camera frame. By rewriting the plane constraint as $\frac{1}{d}\boldsymbol{N}^T\boldsymbol{X} = 1$, eq. (1) becomes:

$$\dot{\boldsymbol{X}} = \hat{\boldsymbol{\omega}}\boldsymbol{X} + \boldsymbol{v}\frac{1}{d}\boldsymbol{N}^T\boldsymbol{X} = \left(\hat{\boldsymbol{\omega}} + \frac{1}{d}\boldsymbol{v}\boldsymbol{N}^T\right)\boldsymbol{X} = \boldsymbol{H}\boldsymbol{X} \tag{2}$$

$\boldsymbol{H} \in \mathbb{R}^{3\times3}$ is known as *continuous homography matrix*. Note that $\boldsymbol{H}$ encodes both the camera linear/angular velocity $(\boldsymbol{v}, \boldsymbol{\omega})$, and the scene structure $(\boldsymbol{N}, d)$.

We define $\lambda\boldsymbol{x} = \boldsymbol{X}$ for a scalar depth factor λ. Note that, by definition, $\dot{\boldsymbol{x}} = \boldsymbol{u}$ which implies $\dot{\boldsymbol{X}} = \dot{\lambda}\boldsymbol{x} + \lambda\boldsymbol{u}$. By using (2), we then get:

$$\boldsymbol{u} = \boldsymbol{H}\boldsymbol{x} - \frac{\dot{\lambda}}{\lambda}\boldsymbol{x}. \tag{3}$$

Here, the depth factor λ can be removed by multiplying both sides of (3) with $\hat{\boldsymbol{x}}$ to obtain the so-called *continuous homography constraint*

$$\hat{\boldsymbol{x}}\boldsymbol{H}\boldsymbol{x} = \hat{\boldsymbol{x}}\boldsymbol{u} \tag{4}$$

since $\hat{\boldsymbol{v}}s\boldsymbol{v} = 0$ for any vector $\boldsymbol{v}$ and scalar s.

B. Classical 4-point algorithm

In order to retrieve $\boldsymbol{H}$, we can stack the elements of $\boldsymbol{H}$ into the vector $\boldsymbol{H}^S = [H_{11}, H_{21}, \cdots, H_{33}] \in \mathbb{R}^9$ and reformulate (4) as

$$\boldsymbol{a}^T\boldsymbol{H}^S = \hat{\boldsymbol{x}}\boldsymbol{u} \tag{5}$$

where $\boldsymbol{a} \in \mathbb{R}^{9\times3}$ denotes the Kronecker product $\boldsymbol{x} \otimes \hat{\boldsymbol{x}}$. This allows us to stack all $\boldsymbol{a}_i$ obtained from n tracked

features into one cumulative matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n]^T \in \mathbb{R}^{3n\times 9}$. Similarly, we stack all $\hat{\boldsymbol{x}}_i\boldsymbol{u}_i$ into a matrix $\boldsymbol{B} = [\hat{\boldsymbol{x}}_1\boldsymbol{u}_1, \cdots, \hat{\boldsymbol{x}}_n\boldsymbol{u}_n]^T \in \mathbb{R}^{3n}$ and extend (5) to the case of n features:

$$\boldsymbol{A}\boldsymbol{H}^S = \boldsymbol{B}. \tag{6}$$

By exploiting standard results, it is possible to recover $\boldsymbol{H}$ from eq. (6) [15]. Since the rank of $\hat{\boldsymbol{x}}$ is 2, at least 4 different feature pairs $(\boldsymbol{x}, \boldsymbol{u})$ are needed to build a matrix $\boldsymbol{A}$ of rank 8 which is the condition to uniqely recover $\boldsymbol{H}$ from (6).

After recovering $\boldsymbol{H}$, it is further possible to decompose it into the triple $(\hat{\boldsymbol{\omega}}, \frac{\boldsymbol{v}}{d}, \boldsymbol{N})$. However, one can prove that, in general, two physically equivalent solutions are compatible with a given homography constraint [15].

C. Case of known camera angular velocity

Gyroscopes directly provide an measurement $\boldsymbol{\omega}_{IMU}$ of the camera angular velocity $\boldsymbol{\omega}$ which can be used to derotate the perceived optical flow field. Therefore, we can subtract the rotational components from the measured flow by exploiting the interaction matrix relating $\boldsymbol{u}$ to $(\boldsymbol{v}, \boldsymbol{\omega})$ [16]:

$$\begin{bmatrix}\boldsymbol{u}'_x\\ \boldsymbol{u}'_y\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}_x\\ \boldsymbol{u}_y\end{bmatrix} - \begin{bmatrix}-x_x x_y & 1+x_x^2 & -x_y\\ -(1+x_y)^2 & x_x x_y & x_x\end{bmatrix}\boldsymbol{\omega} \tag{7}$$

The resulting flow $(\boldsymbol{u}'_x, \boldsymbol{u}'_y)$ does not contain any angular velocity component, thus $\boldsymbol{H}$ can be simplified into

$$\boldsymbol{H} = \frac{1}{d}\boldsymbol{v}\boldsymbol{N}^T. \tag{8}$$

Since $\boldsymbol{N}$ spans $\boldsymbol{H}^T$ and $\|\boldsymbol{N}\| = 1$, it can be retrieved from the singular value decomposition $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ as the first column of matrix $\boldsymbol{V}$. The inherent sign ambiguity can always be resolved by $\boldsymbol{N}_z > 0$. Having retrieved $\boldsymbol{N}$, we can find $\frac{\boldsymbol{v}}{d}$ as $\frac{\boldsymbol{v}}{d} = \boldsymbol{H}\boldsymbol{N}$.

As opposed to the original algorithm, this case leaves us with only one unique solution if at least three feature pairs $(\boldsymbol{x}, \boldsymbol{u})$ are available.

D. Case of known angular velocity and ground plane normal

In our setting, the normal $\boldsymbol{N}$ is assumed to be approximately parallel to the gravity vector. Thus, we can use the ability of onboard IMUs to sense the local gravity vector in order to retrieve $\boldsymbol{N} = \boldsymbol{N}_{IMU}$ by fusing the readings of accelerometers and rate gyroscopes [17].

As a first step, the flow field is derotated as described in eq. (7) such that eq. (8) holds. With $\boldsymbol{N}^T\boldsymbol{x}$ therefore being now a known scalar, we can plug (8) into (4) to obtain

$$\hat{\boldsymbol{x}}\frac{1}{d}\boldsymbol{v} = \frac{\hat{\boldsymbol{x}}\boldsymbol{u}}{\boldsymbol{N}^T\boldsymbol{x}}. \tag{9}$$

Since $\boldsymbol{b} = \frac{\hat{\boldsymbol{x}}\boldsymbol{u}}{\boldsymbol{N}^T\boldsymbol{x}}$ is a known quantity, we obtain a system of three linear equations for each point i in the flow field:

$$\hat{\boldsymbol{x}}_i\frac{1}{d}\boldsymbol{v} = \boldsymbol{b}_i. \tag{10}$$

In order to obtain a least-square approximation of $\frac{\boldsymbol{v}}{d}$ overall n tracked features, we stack all $\hat{\boldsymbol{x}}_i$ into a matrix $\boldsymbol{A} = [\hat{\boldsymbol{x}}_1, \cdots, \hat{\boldsymbol{x}}_n]^T \in \mathbb{R}^{3n\times 3}$ and all $\boldsymbol{b}_i$ into a vector $\boldsymbol{B} = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n]^T \in \mathbb{R}^{3n}$ to obtain the linear system

$$\boldsymbol{A}\frac{\boldsymbol{v}}{d} = \boldsymbol{B} \tag{11}$$

which can be easily solved as $\frac{\boldsymbol{v}}{d} = \boldsymbol{A}^\dagger\boldsymbol{B}$, where $\boldsymbol{A}^\dagger$ denotes the pseudo-inverse of $\boldsymbol{A}$.

Note that any two distinct feature point vectors $\boldsymbol{x}$ will not be parallel due to the perspective projection of the camera. Thus, in principle we need only two flow vectors to compute a solution for $\frac{\boldsymbol{v}}{d}$. However, a more robust estimation is of course obtained by incorporating all the available flow vectors.

III. EXPERIMENTS

A. Experimental setup

For our experiments, we used a quadrotor purchased from MikroKopter.de[2] with a customized low level controller. The vehicle was equipped with a small Q7 board holding an Intel Atom 1.6 GHz CPU and running 11.04. Ubuntu Server. ROS (Robot Operating System) formed a middle layer to allow interprocess communication. The visual input was provided by an IDS[3] uEye UI-1226LE monochrome 752×480 camera. The installed $140°$ lens projects an effective field of view of $102°\times59°$ onto the $1/4''$ sensor. In order to save computation time, the calibration of the visual system was done before the experiments using the Camera Calibration Toolbox for Matlab[4]. We numerically calculated a lookup table which allows the mapping of each pixel on the image plane to the corresponding 3D coordinate $\boldsymbol{x} = [x, y, 1]^T$ in camera frame. As for the onboard IMU, we used a MicroStrain[5] 3DM-GX3 IMU at 250 Hz.

Figures 1a and 1b show the final setup, while Fig. 1c illustrates the different sensor frames used in our setup. All measurements were always converted into the camera frame where all computations were carried out.

To compute the optical flow, we made use of well established and efficient methods provided with OpenCV[6]. In particular, we collected an initial set of Shi-Tomasi features [18] which were then tracked over time using the pyramidal version [19] of the Lukas-Kanade tracker [20]. The number of maintained features was limited to 100 in order to cap both the computational load of the feature tracker and the size of the data package to be processed by the velocity estimator. Whenever the size of the feature set dropped below 50 frames or most of the features were in the outer sides of the image, the set of features was refilled with newly detected features. Since the floor of our flight arena is covered with uniformly white foam, we placed structured carpets on the ground to provide a normal level of features. Lighting was provided by means of default ceiling lights only.

[2] www.MikroKopter.de
[3] www.ids-imaging.com
[4] www.vision.caltech.edu/bouguetj/calib_doc/
[5] www.microstrain.com
[6] opencv.willowgarage.com

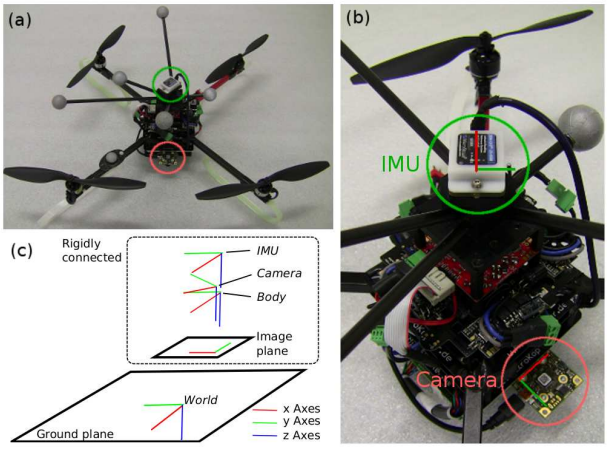


Fig. 1: Experimental setup: (a) Quadrotor in its flight configuration with (b) IMU and camera highlighted. (c) Different sensor and body frames used in this project. The body frame of the quadrotor is fixed in the center of the two crossbars holding the motors. The x-axis equals the red arm. The IMU is aligned with the body frame while the camera is rotated around the z-axis by $45°$ with respect to these frames. The image plane is fixed in $1\ m$ distance to the camera. All frames follow the NED convention commonly used in aviation.

After the velocities were estimated as described in Section II, they were rotated into the quadrotor body frame and used to regulate its velocity. As ground truth, a Vicon[7] setup consisting of six Bonita cameras was used to monitor our $6 \times 8 \times 3$ m flight arena with an accuracy of approximately 0.5 mm. The linear velocities calculated from the tracking data were filtered using a low-pass filter with a cut-off frequency of 10 Hz. After the take-off, which was, together with the landing, done in position control mode using the Vicon system, we switched to pure visual control at 1 m height.

We made use of a gamepad in order to allow a human operator to control the velocity of the UAV. For reasons of simplicity and to allow for wireless flight, this gamepad was connected to a designated ground station. This ground station received the computed velocities from the onboard visual system and generated the appropriate commands for the quadrotor. However, the extremely low computational footprint of these computations would allow for an easy integration into the flying platform at any time. Communication between the visual system and the ground station was carried out by means of WiFi for the velocity estimates while XBEEs[8] were used to transmit the control commands back to the quadrotor.

During the setup of the experiments, we found a strong rotational vibration around the z axis of our quadrotor. Thus, we implemented a 10 Hz low-pass filter on the yaw readings of the gyroscopes as well.

[7] www.vicon.com
[8] www.digi.com

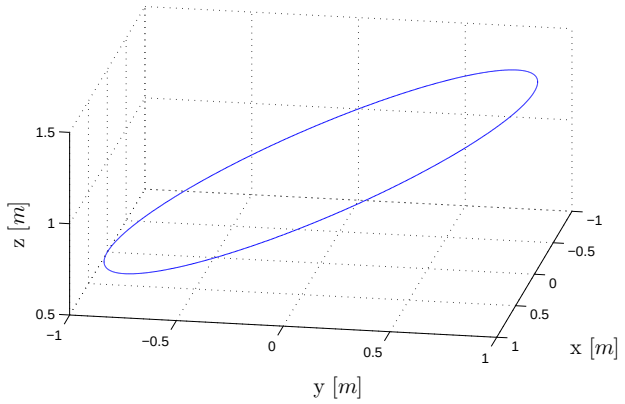


Fig. 2: Circular trajectory of the UAV used for comparison of our three algorithms against the ground truth. The shape of the trajectory measures $2\ m$ in diameter and varies from 0.5 to $1.5\ m$ in height. Additionally, the UAV rotates between $-70°$ in $0.5\ m$ height and $70°$ at $1.5\ m$ around the z-axis. In $10\ s$ the trajectory is traveled once.

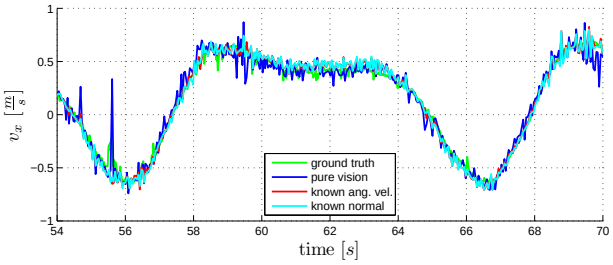
B. Conducted experiments

In order to prove the quality and robustness of our approach, we carried out two sets of experiments. First, we flew the quadrotor along a circular trajectory as depicted in Fig. 2 in open loop control using the Vicon setup as position feedback source. To test the limitations of our approach and to allow for comparison, we ran the three different algorithms consecutively on the same predefined trajectory of $2\ m$ in diameter. One loop was traveled in 10 seconds, thus the vehicle reached a linear velocity of $0.63\ \frac{m}{s}$ in the xy-directions. Additionally, both the height and yaw of the vehicle were varied using a sinusoidal function with the same period of $10\ s$. While the height of the vehicle was kept in the interval of $[0.5 \cdots 1.5]\ m$, the yaw was varied within $[-70 \cdots 70]$ degrees. In particular, this trajectory was chosen to exceed the velocities typically seen in autonomously flying vehicles. For this initial experiment, we used default desktop hardware (2 GHz dual core, 3 GB RAM) for the computation and transmitted all sensory data along cables to minimize lag introduced by the transmission channel. The higher frame rate achieved on this system allows for a sounder comparison of our algorithms.

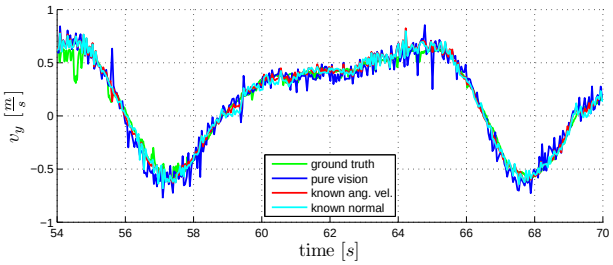
As a second set of experiments, we ported the algorithm to the on-board hardware, closed the control loop and used the sole visual system to control the UAV velocity online. First, we tested the approach by commanding nonzero desired velocity setpoints using the gamepad in order to move the quadrotor within the arena. Then, by commanding a zero velocity, we measured the (unavoidable) drift of the system. An additional low-pass filter was used to filter the linear velocity estimates before they were sent to the controller.

IV. RESULTS AND DISCUSSION

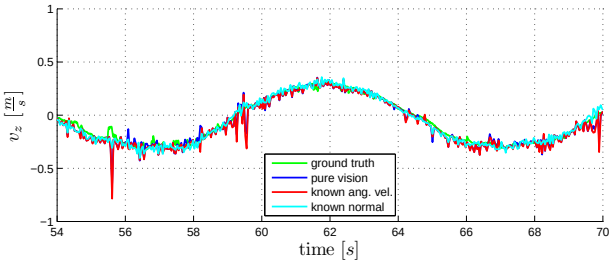
During the first set of experiments performed on desktop hardware, we recovered linear velocities as shown in



(a)

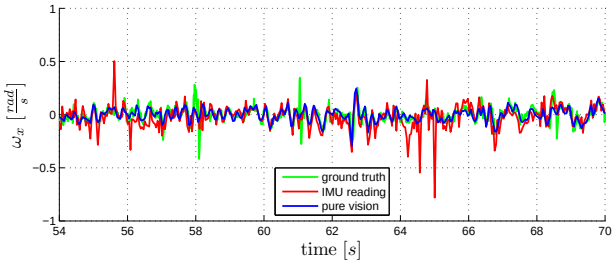


(b)

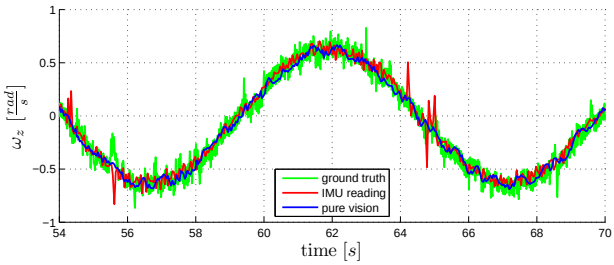


(c)

Fig. 3: Estimated linear velocities in camera frame. (a) Linear velocity along the x, (b) y and (c) z axis. Note that the estimated velocities were scaled by $1/d$ as obtained from the Vicon system for the preparation of this plot.



(a)



(b)

Fig. 4: Estimated angular velocity in camera frame. (a) Angular velocity around the x and (b) z axis. The plot of the velocities around the y axis is very similar to the one around the x axis and was omitted.

Fig. 3(a-c). The retrieved angular velocities are shown in Fig. 4(a-b). Each plot shows a comparison of our three algorithms together with a ground truth in one dimension. Note that IMU readings were used for the angular velocities in case of the two modified algorithms of Section II-C and II-D. Also, as mentioned above, the vibrations present in yaw rotations were effectively filtered for the IMU readings.

Figure 5(a-b) shows the norm error between the estimated linear and angular velocities versus the Vicon ground truth. For the pure visual algorithm, we found a mean error of 0.134 $\frac{m}{s}$ with a standard deviation of 0.094 $\frac{m}{s}$ on the linear velocities. For the other two algorithms, we found slightly better mean errors of 0.117 and 0.113 $\frac{m}{s}$ with a standard deviation of 0.093 and 0.088 $\frac{m}{s}$, respectively. While the incorporation of IMU readings did not greatly improve the quality of the algorithms for linear velocities, the use of IMU readings for the angular velocities does improve the overall performance of the system. The mean error drops from 0.151 $\frac{rad}{s}$ in the case of the pure visual approach to 0.097 $\frac{rad}{s}$ for the IMU readings. The same holds for the standard deviations of 0.110 versus 0.065 $\frac{rad}{s}$. Note, however, that the incorporation of IMU readings mainly improved the estimation of the linear velocity in the x and y axes while there was no significant effect on the z axis.

Figure 5(c) visualizes the origin of the irregularities within the error plots. Since the vehicle is moving with a constant speed in world frame, $\frac{v}{d}$ increases with low altitudes. Thus, the algorithm is forced to sample new features more often since they vanish from the field of view faster. This both slows down the algorithm due to the many resampling processes and does only allow for a sets of 30 to 65 feature pairs instead of the intended 100.

For the analysis of the obtained results in closed loop control, we considered the estimated linear velocity, since this is the only needed information. The used controller is based on standard cartesian trajectory tracking controllers for UAVs [21]. Both the filtered and unfiltered output of our algorithm are shown in comparison with the ground truth in Fig. 6(a-c). Figure 6(d-e) presents the norm of the error between ground truth and our velocity estimate in all three used dimensions. In the closed loop system, we measured a mean error of 0.084 $\frac{m}{s}$ with a standard deviation of 0.139 $\frac{m}{s}$ for the pure visual system. Here, the use of the gyroscopes leads to a significantly lower mean error of 0.039 $\frac{m}{s}$ and 0.042 $\frac{m}{s}$ when knowledge about the plane normal is used additionally. The standard deviation was 0.028 and 0.031 $\frac{m}{s}$, respectively.

First, we commanded the vehicle to move around the arena. The plots in Fig. 6(a-c) present portions of this test. From these plots, one can also estimate the lag introduced by the processing within the visual system and the different filtering steps to approximately 150 ms. Additionally, we

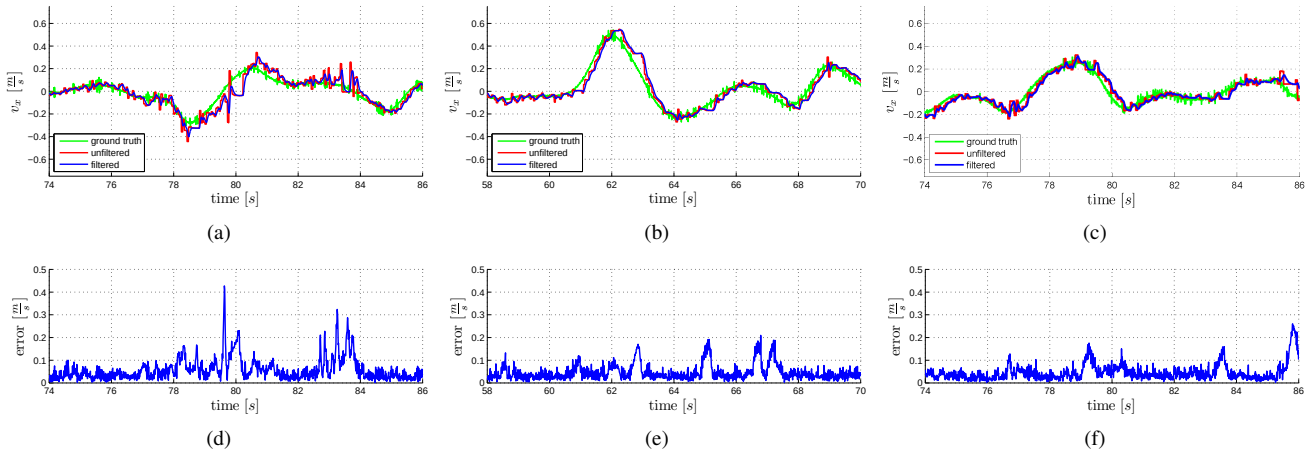


Fig. 6: Linear velocity as estimated on the UAV during closed loop control by our algorithms (filtered and unfiltered) versus ground truth in world frame, together with the norm error. (a) Pure visually estimated x component of the linear velocity, (b) incorporation of angular velocities from the IMU, (c) additional exploit of a known plane normal. Plots for y and z dimension were omitted due to similarity with the x dimension. (d) Norm error between filtered output and ground truth for the pure visual, (e) incorporation of angular velocities and (f) use of an estimated plane normal. The error plots (d) to (f) were compensated for a lag of 150 ms as visible in plot (a) to (c).

also estimated the drift of the system, which was in average about 6 m per minute in the pure visual case and 4 m in the other two cases relying on the IMU. The overall processed frame rate of the system varied between 17 Hz and 18 Hz. While the described latencies together with low controller gains limited the agility of the system, we were still able to actively command the vehicle.

To allow for a comparison with the available ground truth, we solved the height ambiguity for the preparation of Fig. 3 and Fig. 6 using data from the Vicon system. The linear velocities used by the controller, however, were computed using *solely* IMU and camera data, and were only afterwards scaled by the exploiting knowledge of the current height from the ground plane.

V. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In the presented work, we were able to implement an algorithm for closed loop control by means of a single camera for velocity estimation and on-board hardware only. To achieve this, we made use of the continuous homography constraint to recover visual velocity estimation in real-time. The incorporation of angular velocity measurements from gyroscopes led to significantly improved results. However, the additional integration of orientation readings which were used to retrieve the normal vector of the ground plane decreased the tracking error only slightly.

Our approach proved to be a reliable fall back solution for other tracking approaches since it does not rely on maps or on the visibility of special features. While the presence of gyroscopes does lead to a greatly improved performance in closed loop control, the presence of an orientation estimate does not improve the tracking. Thus we were able to present an emergency state for any UAV regardless of the availability of IMU readings from the high level controller or the ability to recover a metric scale. Additionally, we were able to demonstrate the possibility to command the vehicle in such emergency situations as long as some input device is present.

B. Future work

In our next project, we will address two limitations of the presented work concerning indoor environments. First, we will additionally fuse the IMU with the visual system in order to retrieve a metric scale as demonstrated in the literature [22]. Secondly, we will not impose the planar ground plane hypothesis anymore. Thus, this will allow to recover the actual depth of a generic observed scene. Using a wide angle lens, the outer regions of the image can be inspected for clues which indicate vertical obstacles as walls. Additionally, we aim to investigate other more sophisticated ways to retrieve optical flow considering a motion model of the UAV.

VI. ACKNOWLEDGMENTS

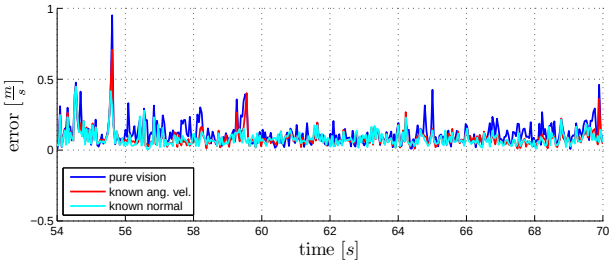
The authors like to thank Dr. Antonio Franchi for his valuable suggestions and contribution on the development of the control framework.

This research was partly supported by WCU (World Class University) program funded by the Ministry of Education, Science and Technology through the National Research Foundation of Korea (R31-10008).

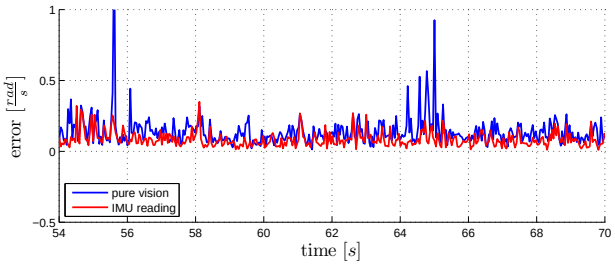
REFERENCES

[1] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic Take Off , Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 221–238, 2010.

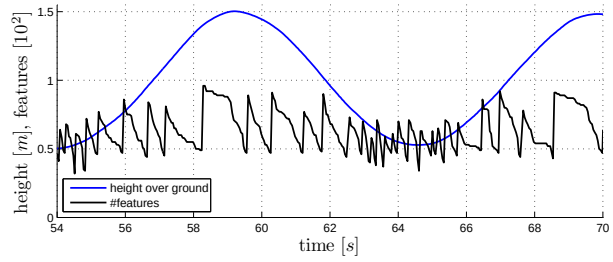
[2] W. Li, T. Zhang, and K. Kühnlenz, "A Vision-Guided Autonomous Quadrotor in An Air-Ground Multi-Robot System," in *Proceedings of the International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 2980–2985.



(a)



(b)



(c)

Fig. 5: Norm of the error between ground truth and both the estimated linear and angular velocities, together with altitude of the vehicle and the number of tracked features: this latter quantity influences the quality of the tracking system. (a) Norm of the error between ground truth and linear and (b) angular velocity estimates. (c) A low altitude of the UAV induces more often the sampling of new features and thus increases the error.

[3] R. A. Newcombe and A. J. Davison, "Live Dense Reconstruction with a Single Moving Camera," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, San Francisco, CA , USA, 2010, pp. 1498–1505.

[4] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007, pp. 225–234.

[5] R. O. Castle, G. Klein, and D. W. Murray, "Video-rate Localization in Multiple Maps for Wearable Augmented Reality," in *Proceedings of the International Symposium on Wearable Computers*, Pittsburgh, PA, USA, 2008, pp. 15–22.

[6] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision Based MAV Navigation in Unknown and Unstructured Environments," in *Proceedings of the International Conference on Robotics and Automation*, Anchorage, AK, USA, 2010, pp. 21–28.

[7] S. Shen, N. Michael, and V. Kumar, "Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV," in *Proceedings of the International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 20–25.

[8] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments," in *Proceedings of the International Conference on Robotics and Automation*, 2011.

[9] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A System for Autonomous Flight using Onboard Computer Vision," in *Proceedings of the International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 2992–2997.

[10] T. Zhang, Y. Kang, M. Achtelik, K. Kuenhnlenz, and M. Buss, "Autonomous hovering of a vision/IMU guided quadrotor," in *Proceedings of the International Conference on Mechatronics and Automation*, Aug. 2009, pp. 2870–2875.

[11] R. Mahony, F. Schill, P. Corke, and Y. S. Oh, "A new framework for force feedback teleoperation of robotic vehicles based on optical flow," in *Proceedings of the International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 1079–1085.

[12] L. R. García Carrillo, A. Dzul, R. Lozano, and C. Pégard, "Combining Stereo Vision and Inertial Navigation System for a Quad-Rotor UAV," *Journal of Intelligent & Robotic Systems*, 2011.

[13] F. Kendoul, I. Fantoni, and K. Nonami, "Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles," *Robotics and Autonomous Systems*, vol. 57, no. 6-7, pp. 591–602, June 2009.

[14] T. Y. Tian, C. Tomasi, and D. J. Heeger, "Comparison of Approaches to Egomotion Computation," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, vol. 0, no. 2, San Francisco, CA , USA, 1996, pp. 315–320.

[15] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision*. Springer, 2004.

[16] F. Chaumette and S. Hutchinson, "Visual Servo Control, Part I. Basic Approaches," *Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[17] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[18] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, no. June, Seattle, WA, USA, 1994, pp. 593–600.

[19] J.-Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm," *In Practice*, vol. 1, no. 2, pp. 1–9, 1999.

[20] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Vancouver, BC, Canada, 1981, pp. 674–679.

[21] S. Bouabdallah and R. Siegwart, "Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor," in *Proceedings of the International Conference on Robotics and Automation*, no. April, Barcelona, Spain, 2005, pp. 2247–2252.

[22] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM," in *Proceedings of the International Conference on Unmanned Aerial Vehicles*, Dubai, United Arab Emirates, 2010.