

PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision

Lorenz Meier · Petri Tanskanen · Lionel Heng ·
Gim Hee Lee · Friedrich Fraundorfer · Marc Pollefeys

Received: 29 July 2011 / Accepted: 30 January 2012 / Published online: 23 February 2012
© Springer Science+Business Media, LLC 2012

Abstract We describe a novel quadrotor Micro Air Vehicle (MAV) system that is designed to use computer vision algorithms within the flight control loop. The main contribution is a MAV system that is able to run both the vision-based flight control and stereo-vision-based obstacle detection parallelly on an embedded computer onboard the MAV. The system design features the integration of a powerful onboard computer and the synchronization of IMU-Vision measurements by hardware timestamping which allows tight integration of IMU measurements into the computer vision pipeline. We evaluate the accuracy of marker-based visual pose estimation for flight control and demonstrate marker-based autonomous flight including obstacle detection using stereo vision. We also show the benefits of our IMU-Vision synchronization for egomotion estimation in additional experiments where we use the synchronized measurements for pose estimation using the 2pt+gravity formulation of the PnP problem.

Keywords Micro aerial vehicles · Quadrotor · Computer vision · Stereo vision

1 Introduction

We introduce a novel quadrotor MAV system, the PIXHAWK MAV (Fig. 1), which is specifically designed to be

a research platform for computer vision based flight control and autonomous flight using computer vision. The main contribution is a MAV system that is able to run both the vision based flight control and stereo vision based obstacle detection parallelly on an embedded computer onboard the MAV. The system integrates a computing board that is powerful enough to handle all image processing and flight control processes onboard onto a small scale quadrotor MAV. The proposed flying system carries up to $6\times$ the computation power of comparable systems of the same size, e.g. Achtelik et al. (2011). With the possibility of performing all computational processes onboard without the requirement for a constant data link to a ground station, our design brings the vision of a fully autonomous quadrotor MAV significantly closer.

A key feature of our system is the hardware IMU-camera synchronization. This allows us to measure the USB image transmission delays in our system precisely. As a result, we are able to do visual pose estimation with the synchronized IMU measurements with improved efficiency and robustness. This algorithm is evaluated and compared to a vision only marker based pose estimation algorithm. A stereo vision based obstacle detection system is integrated onto our MAV system. The stereo computer vision system produces a depth map that gives detailed information about the obstacle as compared to other sensors such as infrared or sonar.

We demonstrate the capabilities of the system in the conducted experiments. We perform autonomous waypoint based flights using only vision and compare the accuracy of the vision pose estimation with Vicon groundtruth. In addition, we compare the vision-only pose estimation to a combined IMU-Vision pose estimation using the 2pt+gravity PnP algorithm (Kukelova et al. 2010). Lastly, we show the functionalities of our stereo vision obstacle detection module.

Electronic supplementary material The online version of this article (doi:10.1007/s10514-012-9281-4) contains supplementary material, which is available to authorized users.

L. Meier (✉) · P. Tanskanen · L. Heng · G.H. Lee ·
F. Fraundorfer · M. Pollefeys
ETH Zurich, CAB G 86.3, Universitaetstr. 6, 8092 Zurich,
Switzerland
e-mail: lm@inf.ethz.ch

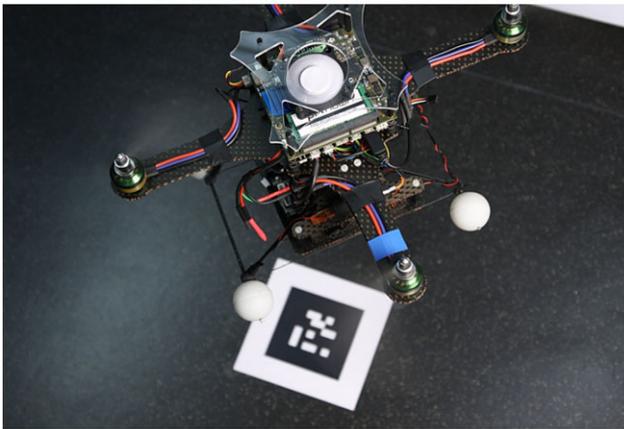


Fig. 1 PIXHAWK Cheetah Quadrotor

All our hardware and software designs are made open-source and are published on our web page¹ with the goal to create an open research platform for the community.

1.1 State-of-the-art

This paper extends our previous work described in Meier et al. (2011). We give more detailed descriptions and evaluations of the system, in particular the analysis of the USB image transmission delays. We also add a comparison of a combined IMU-vision pose estimation algorithm (2pt+gravity (Kukelova et al. 2010)) to vision-only pose estimation. In addition, we show fully autonomous onboard computer vision based waypoint navigation using visual markers and demonstrate the integrated vision based obstacle avoidance system.

Much of the previous research in autonomous unmanned aerial vehicles (UAVs) has been based on large UAVs in the weight range of 10–20 kg. UAVs of this size are able to carry an extensive sensor suite, e.g. LIDAR, Radar, camera system and powerful onboard computers. Impressive results have been shown in terms of autonomous take-off, landing and navigation as well as obstacle avoidance (Conte and Doherty 2008; Saripalli et al. 2002; Johnson et al. 2005; Bosch et al. 2006; Scherer et al. 2008; Proctor et al. 2006; Kanade et al. 2004; Hrabar and Sukhatme 2009). Specific adaptations to the algorithms and sensor hardware are needed to apply these results to small scale MAVs under 1.5 kg.

Recent works successfully demonstrated the use of small LIDAR sensors on such small scale MAVs for mapping and autonomous flight (Hofmann et al. 2004; Roy et al. 2010; Dryanovski et al. 2011; Shen et al. 2011; Bachrach et al. 2010; Fowers et al. 2007). However, pure vision based autonomous flight control and mapping for small scale MAVs has yet to reach the same level of maturity as with LIDAR

sensors. One of the first works in visual MAV control was done by Kemp (2006). He used an a-priori generated 3D model of the flight area and 2D-3D edge matching to compute the MAV pose. He demonstrated on-spot hovering of an MAV. The MAV only carried a small analog camera with wireless image transmission. All processings were done off-board.

More recently, Blösch et al. (2010) described visual autonomous flight using an Asctec Hummingbird and a downward looking camera. A visual SLAM algorithm was running off-board on a standard PC. The images of the on-board camera are streamed to the PC using a USB cable physically connected to the MAV thus limiting the autonomy. Control input was sent back to the MAV via a radio link. In their paper, they demonstrated on-spot hovering and waypoint following over a 10 m trajectory. To ensure enough visual features for SLAM, their testbed was covered with textured posters. The off-board visual SLAM computed positions at varying frame rates between 15–30 Hz. An extension of Blösch et al. (2010) was described by Achtelik et al. (2011). They replaced the Asctec Hummingbird with a bigger model, the Asctec Pelican, and equipped it with an Intel Atom onboard computer. This allowed them to run a modified version of the visual SLAM of Blösch et al. (2010) on-board with a frame rate of 10 Hz. However, this modification severely limits the size of the environment that can be mapped. They also described a position controller and successfully demonstrated closed-loop position control with only visual feedbacks. In addition, they demonstrated on-spot hovering in an indoor and outdoor setting. However, the vision based flight control used up all the processing power of the embedded computer, leaving none for the other processes, e.g. obstacle avoidance.

Williams et al. (2011) used line and point features for visual flight control of a MAV. They described three types of flight patterns: traversing, hovering and ingress. In their experiments, they computed the MAV trajectory offline using previously captured images on a desktop PC for the different flight patterns.

An approach for higher level navigation implemented on a Parrot AR.Drone was described by Bills et al. (2011). The commercially available Parrot AR.Drone comes with a forward and downward looking camera and the capability of onboard on-spot hovering making the system easy to use. However, the system is closed and has no payload capability. It is only possible to stream the images of the camera using Wi-Fi and control it with Wi-Fi. In their work, Bills et al. controlled the direction of movement of the MAV from perspective cues obtained from images and from classification of the environment. This allowed the MAV to follow corridors and even make turns. However, there was no notion of a metric map and the image processing is completely done off-board on a desktop PC.

¹ www.pixhawk.ethz.ch.

The image processing can be greatly simplified with the use of artificial markers. Artificial markers were used by Eberli et al. (2011) for hovering, take-off, and landing. They described the use of one circular marker to compute the position and pose of the MAV. In their experiments, the MAV was connected via USB cable to a ground station and the image processing was done off-board on a desktop PC.

A different approach by Li et al. (2011) showed hovering, take-off and landing of an Asctec Hummingbird equipped with an Intel Atom onboard computer. They described the use of an active LED marker pattern. In their approach, the flight control was done on-board. They demonstrated hovering over a marker pattern which was mounted on top of a ground robot and as a result, the MAV was able to follow the trajectory of the ground robot.

A similar approach was described by Wenzel et al. (2011). They demonstrated hovering, take-off, and landing of an Asctec Hummingbird using a marker platform mounted on top of a ground robot. The marker pattern was made of IR LED's and the MAV's position was computed from a Wii-mote sensor fitted to the MAV. The Wii-mote sensor performs hardware image processing and directly outputs the point coordinates of the detected pattern. The final pose computation was then directly done on the low-level controller of the MAV.

Substantial existing research relies on outside-in-tracking of the MAV, e.g. by means of a Vicon motion capture system, to measure the vehicle position (Heng et al. 2011; Mellinger and Kumar 2011; Mellinger et al. 2010; Ducard and D'Andrea 2009). These works mainly focus on low-level control problems or higher-level tasks assuming prior knowledge of the MAV positions and use off-board control.

The PIXHAWK MAV system design itself is an alternative to commercially available MAVs, such as Asctec MAVs,² MicroKopter,³ MicroDrones⁴ or Parrot AR.Drone.⁵ The hardware design is quite similar to commercially available MAVs. However, while most of the systems have a closed control architecture, our system is primarily designed as a research platform, and therefore, has an open control architecture that provides easy access to all the low level measurements and readily accepts control inputs from higher-level on-board computers. In contrast to the commercial products, our system is an open source and open hardware design, which allows researchers to adapt every single detail as needed. Many commercially open-source systems either only allow users to modify a part of the software or to replace the complete software stack or explicitly do not allow the modification and reuse. This requires users to completely

rebuild the whole software stack, while our approach is to allow incremental improvements and specializations based on an initial state. Together with the software architecture, the ground control and operator software, and the easy to use marker based localization, the PIXHAWK system is a great testbed for MAV research.

1.1.1 Comparison of PIXHAWK quadrotor platform (mid 2011)

Table 1 shows the difference between different quadrotor platforms. While the autopilot capabilities of the PIXHAWK quadrotor are similar to another competitive systems, the on-board computational speed, RAM and solid-state disk interfaces are unmatched in the MAV domain. A “–” stands for not present, a “x” for present and “o” for partially present. While the power consumption of the Core 2 Duo onboard computer is substantially higher than the alternatives, its performance is up to 6× better than the Atom computer which consumes 11 W. The maximum consumption of 27 W does not substantially contribute to the overall consumption of the quadrotor, which is about 150–180 W only for the motors and about 200 W in total with all electronics and onboard computer.

1.1.2 Comparison of MAVCONN middleware in MAV-specific features

Table 2 compares different middlewares. As MAVCONN uses internally LCM as transport layer, all features available in LCM are retained in MAVCONN. As LCM is a transport layer and not a full robotic middleware, it optimally combines with MAVCONN to a full solution for MAVs. Our middleware adds a layer on top of LCM, providing the MAVLink message format and interfaces to peripherals such as radio modems or USB machine vision cameras. The main difference to ROS is the distributed communication model without central server and the capability to fully communicate over radio links when needed. No message rewriting is necessary to communicate with MAVLink-enabled IMUs or a ground control station. MAVCONN thus resembles the upper layers of ROS, while LCM is a substitute for the ROS communication layers. MAVCONN is intended for a different application scenario as ROS and focuses more on small-scale system where different electronic modules are not interfaced by ethernet (in contrast to large ground robots), but often employ simple RS-232 serial communication. ROS requires in this scenario to rewrite ROS messages in a serial format, while MAVCONN messages can be directly passed between UDP and serial transport layers. This allows to use one generic bridge process to bridge between different transport media, in contrast to ROS which requires one customized bridge process per type of serial peripheral.

²www.asctec.de.

³www.mikrokopter.de.

⁴www.microdrones.com.

⁵ardrone.parrot.com.

Table 1 Comparison of PIXHAWK hardware platform with common systems

System	PIXHAWK	Asct. Pelican	AR.Drone	Mikrokopt.	Arducopt.
CPU	CORE 2 Duo	Intel Atom	ARM9	ARM9	–
CPU cores	2	1	1	1	–
CPU MHz	1.86 GHz	1.2 GHz	468 MHz	90 MHz	–
RAM	2 GB	1 GB	128 MB	96 KB	–
Peak consumption	27 W	11 W	1 W	0.2 W	–
USB ports	7	7	1	1	–
PCIe ports	0	1	0	0	–
S-ATA ports	1	0	0	0	–
UARTs	4	2	0	2	–
Autopilot	ARM7	ARM7	PIC24	ATMega	ATMega
AP MHz	60 MHz	60 MHz	24 MHz	16 MHz	16 MHz
AP RAM	32 KB	32 KB	8 KB	96 KB	8 KB
3D Gyro	x	x	x	x	x
Accelerometer	x	x	x	x	x
Compass	x	x	–	x	x
Open HW	x	–	–	–	x
Open SW	x	o	o	o	x
Typ. max. weight	1.5 kg	1.5 kg	0.6 kg	1.5 kg	1.5 kg
Typ. prop. diam.	10"	10"	7"	10"	10"

Table 2 Comparison of MAVCONN middleware with LCM and ROS. MAVCONN is based on LCM and adds a top layer specific to MAV applications on it

Software name	MAVCONN	LCM	ROS
Type	MAV Middleware	RT Comm. Library	Robot Middleware
Message format	x	–	x
Ground control avail.	x	–	–
Radio modem support	x	–	–
Serial comm support	x	–	–
UDP support	x	x	x
UDP transport layer	LCM	LCM	ROS
UDP latency	100–1100 μ s	100–1100 μ s	500–1100 μ s
Stereo triggering	x	–	–
IMU sync	x	–	–

Also other management functions, such as the parameter server, are decentralized in MAVCONN and support Linux processes and microcontrollers with the same API.

2 System design

The PIXHAWK design includes a powerful onboard computer which makes it possible to run high-level tasks, in particular visual localization, onboard the MAV. The system design is depicted in Fig. 2. Visual localization, obstacle detection, and planning are implemented on the onboard high-level flight computer, an Intel®Core™2 Duo. The visual localization module computes the full 6-DOF pose of the MAV (see details in Sect. 3). The stereo obstacle detection module computes real-time disparity maps

from the front-looking stereo pair (see Sect. 4). The output of the stereo module can be used for obstacle avoidance by the planning module. The planning module currently implements waypoint following. Both the attitude and position controllers are implemented on a low-level real-time controller (see Sect. 5 for state estimation). The position controller takes as input both the poses from visual localization and the setpoints generated from the planner. For MAV control, the attitude measurements and the vision pose estimates need to be synchronized. In our system, the synchronization is solved by using the IMU to hardware-trigger the cameras and timestamp the measurements.

2.1 Vision-IMU synchronization

Data from different sensors, in particular from multiple digital cameras, is synchronized by an electronic shutter sig-

Fig. 2 The PIXHAWK quadrotor system design. The powerful onboard computer enables high-level tasks such as visual localization, obstacle detection, and planning to run onboard. Position and attitude estimation are implemented on a low-level real-time controller

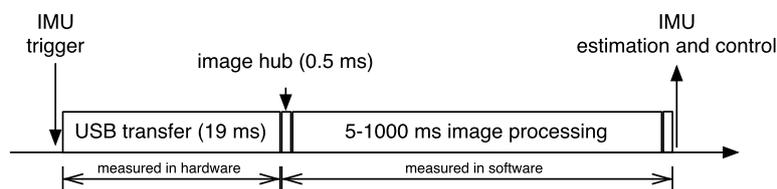
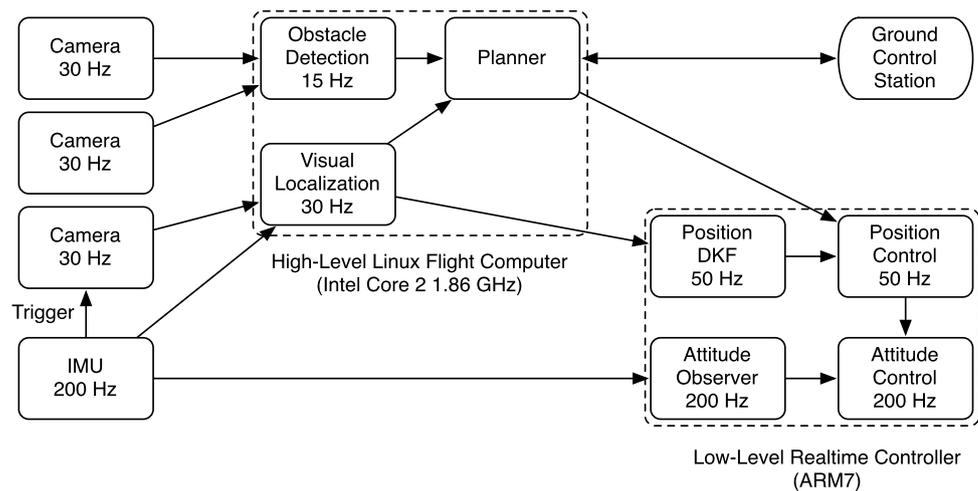


Fig. 3 System delays. Initially, the camera shutters are triggered by the inertial measurement unit. The first 19.5 ms after closing the shutter (outdoors after 0.3 ms, indoors after 2–5 ms) are consumed by the USB transfer of the image from the camera to the volatile memory of the onboard computer. To support multi-process computer vision, the image is transported to different processes via shared memory in RAM

through the image hub software interface of MAVCONN. Processing the image in Linux requires between 5 ms for ARToolkit and several seconds for large-scale localization and mapping approaches. USB and UART transfer delays are only observable with hardware triggering; otherwise, they remain unknown

nal, and assembled into a timestamped sensor message. IMU sensor data, for example, absolute attitude and angular rates, is available as part of the image metadata. Images are transmitted over the USB bus to the camera process, while the IMU measurements and the shutter timestamp are delivered via a serial interface. Image transmission from the cameras to main memory via USB takes approximately 16–19 ms (where the time differs slightly for each image), while the transmission of the shutter timestamp from the IMU to main memory via serial/MAVLink takes approximately 0.1–2 ms. As it is guaranteed that the IMU data arrives earlier than the image, the camera process can always deliver images labeled with IMU metadata via the middleware to subscribing processes. Figure 3 shows the contributions of individual processing steps to the overall system delay. The transfer time of the image content from the camera module over USB 2.0 is substantial where fast localization techniques are concerned.

3 Visual position estimation

As the PIXHAWK middleware provides a precise time base, a standard textbook estimation and control pipeline was

proven to perform well for autonomous flight. Figure 2 shows the localization and control architecture. Images are read from the camera at 30 Hz and the position is estimated at the full camera rate, using additional inertial information. The current position is then used by the onboard mission planner to determine the desired position. The current and the desired positions are fed back to the position estimation and control software module running on the ARM7 autopilot controller.

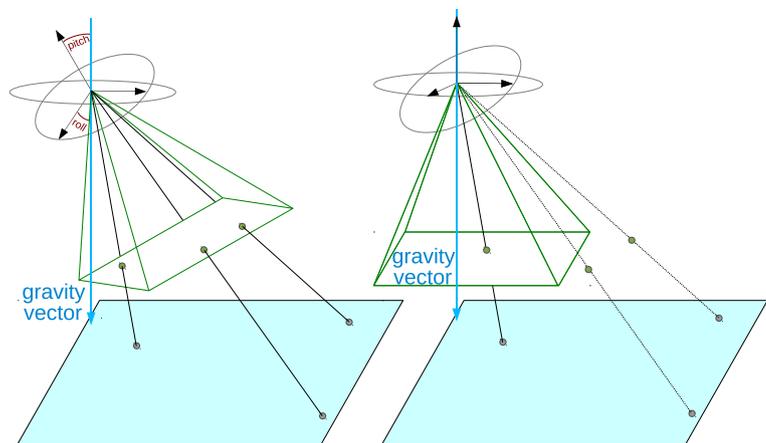
3.1 ARTK+ localization

We use a localization test bed that uses markers with an adapted implementation of ARToolkit+ (Wagner and Schmalstieg 2007) for the localization. The marker positions are encoded in a global world map with the 6D position and orientation of each marker. By extracting the marker quadrangle, the global marker position can be estimated. The correct orientation on the quadrangle plane and the marker ID are encoded by a 2D binary code inside each marker. An example of a larger marker setup is shown in Fig. 4. However, the system itself is not dependent on this particular approach—any kind of localization algorithm can be used. The main benefit of using ARToolkit+ in the test bed

Fig. 4 Flight environment with the ARToolkit markerboard on the floor



Fig. 5 Relation of gravity vector and camera angles. The right-fronto parallel view is obtained by rotating the image plane by the roll and pitch angles



setup is its relatively low delay (5–10 ms), its robustness with respect to suboptimal lighting conditions, and the high robustness to motion blur. It is therefore very suitable for system testing and validation.

3.2 Vision-IMU fusion

The inertial measurement unit can be exploited to increase the robustness and accuracy of the vision-based localization. There are classical IMU-vision fusion methods but all these methods expect covariance estimates of the vision position data, which is prone to errors due to outliers in feature matching. A more promising method is to directly include the IMU data in the vision estimation during the outlier filtering and position estimation.

If the accuracy of the IMU-estimated vertical direction is better than the pure vision estimate, the IMU-estimated vertical direction can be used by the vision algorithm to improve the localization accuracy. Figure 15 shows that the localization accuracy increases when using two points and the known vertical from the inertial data instead of four points.

In the case of the 2-point algorithm (Kukelova et al. 2011), the calculation steps for the pose estimation are significantly simplified when substituting parts of the four-point equation with the IMU roll and pitch (see Fig. 5 for the geometric relation where the image plane is aligned with the surface normal formed by the gravity vector). This speeds up RANSAC which is typically used to filter out outlier matches between image features. Hence, the benefits of directly combining IMU and vision are, depending on the methods used, improved accuracy and computation time.

Since the roll and pitch angles of the camera are known through the inertial measurement unit, the lines connecting the camera center and the 3D points seen by the camera can be rotated to compensate for roll and pitch. The equation for projecting image points into the homogeneous camera space and rotating the rays is given in (1); the same operation is also depicted in Fig. 5. The pixel coordinate u is projected with the inverse camera matrix K^{-1} into the normalized homogeneous coordinate space. This ray is then rotated by the roll and pitch with the rotation matrix formed by multiplying the rotation matrices around the roll and pitch

$R_{\phi\theta}$. The resulting ray in homogeneous coordinates is now fronto-parallel with respect to the ground plane.

$$u' = R_{\phi\theta} K^{-1} u \quad (1)$$

The resulting fronto-parallel view has only x , y and ψ as free parameters. Kukulova et al. provided a closed-form solution to localize from two points and the known vertical direction by solving for x , y , z and ψ . By applying this algorithm in a least squares sense on the ARToolkit correspondences, we obtain the final position output, since ϕ and θ are already known. Figure 15 shows that the solution of the vision-IMU 2-point algorithm outperforms in terms of accuracy the 4-point algorithm used in ARToolkit+. Both algorithms operate on the same set of visual correspondences based on the ARToolkit+ corners.

In addition, for any non-global vision-based localization approach, the IMU information can provide the gravity vector and heading to be used as the global reference. This is especially important for loop closure in SLAM where global attitude information can facilitate loop detection and reduce convergence into local minimas. This also explains the advantage of storing the absolute attitude in image metadata rather than the attitude relative to the previous frame. Using the absolute attitude, it is always possible to extract the relative orientation between any pair of images.

3.3 Outlier removal

The obtained position vector x , y , z and ψ is filtered with a 4×1 D block Kalman filter in the next step; this implies that the filters are parameterized with an error model of the computer vision approach. As IMU and vision both estimate the 3-DOF attitude of the helicopter, this redundant data can be used to detect and remove position outliers produced by the localization step. Any erroneous vision measurement will not only contain a wrong position estimate but also a wrong attitude estimate because of the projective dependency of the position and attitude. Position outliers can therefore be rejected based on the comparison of roll and pitch estimates from the IMU and from the visual localization. Notice the effect on position outliers when including the IMU data into the vision estimation in Fig. 15. The variance of the position estimate is reduced by a significant amount such that there is no need for subsequent outlier removal.

4 Stereo obstacle detection

The front looking stereo camera allows us to get depth information in both indoor and outdoor environments, and with depth information, we can reliably detect obstacles in the MAV's vicinity, and compute their locations. By running

stereo processing algorithms onboard the MAV, we demonstrate the ability of the onboard computer to handle computationally intensive tasks which would otherwise be not possible on typical MAVs equipped with single-core processors. In our stereo processing pipeline, we compute disparity data from stereo image pairs, and subsequently, compute a point cloud which is used to update a 3D occupancy map. We determine a cell to be an obstacle if its occupancy probability exceeds a preset threshold, which in our case, is 0.5. If an obstacle is observed to be within the safety clearance of the MAV, an alert message is published. Any planning module that receives this alert can either perform an emergency stop or take evasive maneuvers.

4.1 Point clouds from stereo

With each stereo image pair, we rectify both images, and use a block-matching stereo correspondence algorithm to build a dense 640×480 disparity map. Subsequently, we compute the depth to the points in the scene relative to the camera coordinate system:

$$z = \frac{bf}{d} \quad (2)$$

where d is the disparity. Differentiation of (2) with respect to d yields:

$$\Delta z = \frac{bf}{d^2} \Delta d \quad (3)$$

Δz denotes the resolution of the range measurement corresponding to d . To avoid spurious range measurements due to small disparities, we set the minimum disparity:

$$d_{min} = \left\lceil \sqrt{\frac{bf \Delta d}{\Delta z}} \right\rceil \quad (4)$$

In our case, we choose conservative values of $\Delta z = 0.25$ and $\Delta d = 0.5$. With these values, the maximum range of our stereo camera with a baseline of 5 cm and a focal length of 645 pixels is 4 m.

We compute the 3D coordinates of each pixel relative to the camera coordinate system:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \frac{z}{f} \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \quad (5)$$

where z is the depth associated with the pixel, (c_x, c_y) are the coordinates of the principal point of the camera, f is the focal length, and (i, j) are the image coordinates of the pixel. The values of c_x , c_y , and f , together with the stereo baseline b are obtained from an one-time calibration.

We then find the world coordinates of each point:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = {}_w^i H_i^c H \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \quad (6)$$

where w_iH is the homogeneous transform from the world frame to the IMU frame and is estimated by the visual odometry, and c_iH is the homogeneous transform from the IMU frame to the camera frame and is estimated using the InerVis calibration toolbox (Lobo and Dias 2007).

5 State estimation and control

The state estimation is run on the low-level controller to satisfy the real-time requirements of the system. In addition, this allows us to compensate for the loss of either visual localization data or the complete onboard computer connection. In the case of such a catastrophic event, the system performs an open-loop safety landing maneuver. To synchronize inertial data and vision estimates, the IMU maintains a buffer of attitude measurements corresponding to the last n image frames. Once the IMU receives a vision position estimate, it reads out the buffered sensor values, and performs a state estimator update.

5.1 Discrete Kalman estimation

After the outlier rejection, the remaining positions are more conformant to the normal distribution, and thus, allow the use of a simple discrete Kalman filter. As the dynamics of a quadrotor are only loosely coupled in the x , y and z directions (Bouabdallah and Siegwart 2007), the dynamics can be modeled as three independent dimensions. As the yaw angle taken from computer vision is of much better accuracy and resolution in indoor settings compared to the yaw angle from a magnetometer due to iron structures in the building, the yaw angle is taken as the fourth independent dimension for filtering. Given this quadrotor dynamic model, the Kalman filter is designed as a block of $4 \times 1D$ Kalman filters with the position and speed as states. The Kalman filter assumes a constant speed model, and takes the position estimate as input. The estimated velocity is critical in damping the system, as the only physical damping is the air resistance on the horizontal plane which is insignificant at the hovering and low-speed conditions the system is typically operating in. The states of the four Kalman filters are:

$$x_k = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad y_k = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} \quad z_k = \begin{bmatrix} z \\ \dot{z} \end{bmatrix} \quad \psi_k = \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix}$$

We estimate the current state of the vehicle x_k which is modeled by

$$x_k = A \cdot x_{k-1} + w_{k-1}$$

Where the dynamics matrix A models the law of motion, x_{k-1} is the previous state and w_{k-1} the process noise.

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

This motion is measured at certain time steps where the measurements are expressed as the gain H times the current state plus the measurement noise v .

$$z_k = H \cdot x_k + v_k$$

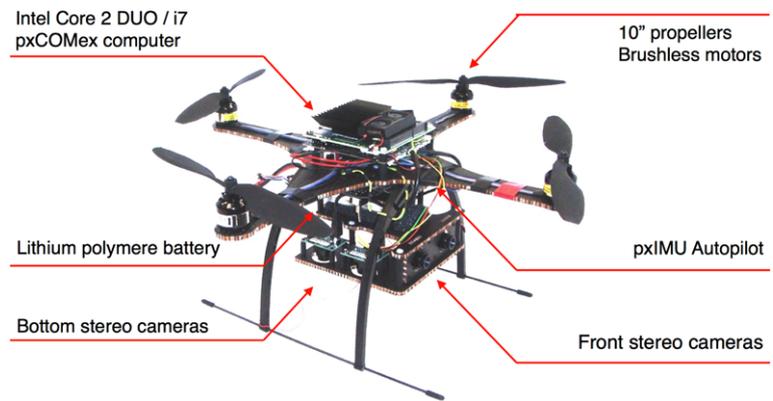
The speed in the model will therefore only be changed by measurements, and assumed constant during prediction. From this formulation, it is already obvious that varying time steps can be handled by the filter as long as they are precisely measured. As this filter does not include the control input matrix B , the filter assumes a constant speed model which is a valid approximation if the filter update frequency is fast enough with respect to the change of speed of the physical object. Because the PIXHAWK system provides a precise time base, the filter uses the measured inter-frame interval as the time difference input δt . If measurements are rejected as outliers, the filter only predicts for this iteration, and compensates in the next update step for the then longer time interval. This allows the system to estimate its egomotion for up to about 500 ms and recover from several dropped camera frames.

5.2 Position and attitude control

The current and the desired positions are fed back to the position estimation and control software module running on the ARM7 autopilot controller. The autopilot calculates the desired attitude, and controls the attitude using its onboard inertial sensor suite. The x - and y -positions are controlled with the angle of attack of the collective thrust vector by setting the desired pitch angle for x and the desired roll angle for y . The z -position can be controlled with the component of the collective thrust collinear to the gravity vector. The yaw angle can finally be controlled by the difference of rotor drag of the clockwise (CW) and counter-clockwise (CCW) rotor pairs. As the discrete Kalman filter contributes a smooth position and speed estimate with little phase delay, the controller can be designed as a standard PID controller implemented as four independent SISO PID controllers for x , y , z , and yaw, as in Bouabdallah et al. (2004). Attitude control is implemented following the standard PID based attitude control approach for quadrotors using one PID controller for each of the roll, pitch, and yaw. The craft is actuated by directly mixing the attitude control output onto the four motors.

6 Micro air vehicle and middleware

Off-board processing effectively makes the MAV dependent on the external processing unit, and severely limits the safety and operation range of the vehicle. Our system brings the multi-process architecture and onboard processing capabilities from the 20–100 kg class to vehicles with around 1.2 kg

Fig. 6 Quadrotor components

liftoff weight. In contrast to systems using local stabilization approaches on specialized microcontroller hardware (Parrot AR.Drone), the system is geared towards global localization and autonomous exploration of unknown environments using stereo vision. The presented initial results show that with a 30 Hz frame rate, our system consumes 10% of the maximum CPU load (5 ms processing time per frame) for autonomous marker based flight and 40–60% load (20 ms processing time per frame) for stereo-based obstacle avoidance, which leaves enough capacity for future work. Since the onboard computer offers two CPU cores, onboard parallel localization and mapping is within reach. GPS and, to a large extent, laser-based systems can offer a deterministic processing time to fuse the sensor data into the localization. In contrast, computer vision has varying, and in comparison, often longer processing times, depending on the image content. Therefore, the estimation and control steps cannot assume a fixed interval length Δt and a fixed processing delay Δp . Instead, they must use the actual timestamp of all measurements to calculate the correct latency. Thus, all data in our system is timestamped with a resolution in the order of microseconds. This data includes images from multiple cameras, the system attitude, acceleration data, and barometric data.

6.1 Stereo head

For the stereo head, the middleware supports 2 Point Grey Firefly MV or MatrixVision Bluefox cameras that respectively capture 640×480 and 752×480 grayscale images at up to 60 Hz. The camera interface allows additional camera models to be supported. However, since there is no standard for trigger-support among different camera module manufacturers, it is necessary to implement a small interface class in the middleware for each camera type. The camera pair is rigidly mounted with a baseline of 5 cm on a carbon composite frame as shown in Fig. 6. The hardware trigger provided by the IMU enables synchronous capture of images from both cameras; this synchronous capture is crucial for accurate estimation of stereo disparity.

6.2 Mechanical structure and flight time

Our custom mechanical design (Fig. 6) effectively protects the onboard processing module in case of a system crash, and the fixed mounting of the four cameras allows inter-camera and camera-IMU calibration. Our system comes in two sizes, one optimized for very small indoor environments, and one standard size. As the processing board and up to four cameras represent a relatively large payload of 400–800 g for the small diameter of 0.55 m (0.70 m for the larger version) of the quadrotor, the overall system structure has been optimized for low weight. It consists of lightweight sandwich material with composite plates and an inner layer made of Kevlar. Each of the four motors with 8" or 10" propeller contributes a maximum of 450–600 g thrust, enabling the system to lift a 400 g payload at a total system weight of 1.0–1.2 kg, including the battery. This allows a continuous flight time of 7–9 minutes with 8" propellers and 14–16 minutes with 10" propellers. The propulsion consumes 150–180 W for hovering, while the onboard computer consumes only 27 W peak. Therefore, flight time is governed by the weight of the system.

6.3 Flight and processing electronics

The PIXHAWK Cheetah quadrotor design was built from scratch for onboard computer vision. With the exception of the commercial off-the-shelf (COTS) motor controllers and cameras, all electronics and the mechanical frame are custom designed (Fig. 7). First, the payload consisting of the pxCOMEx processing module and up to four machine vision cameras (PointGrey Firefly MV USB 2.0 or MatrixVision Bluefox), was selected. The system design then followed the requirements of onboard computer vision. The onboard electronics consists of an inertial measurement unit and autopilot unit, pxIMU, and the onboard computer vision processing unit, pxCOMEx.

Fig. 7 Onboard sensors and avionics with electronic buses

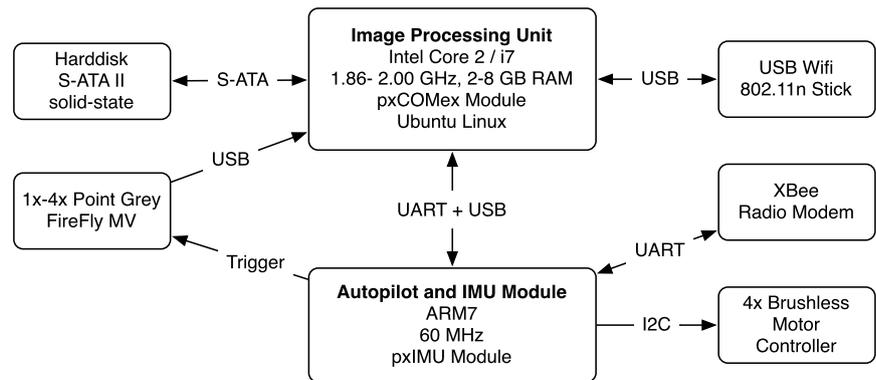
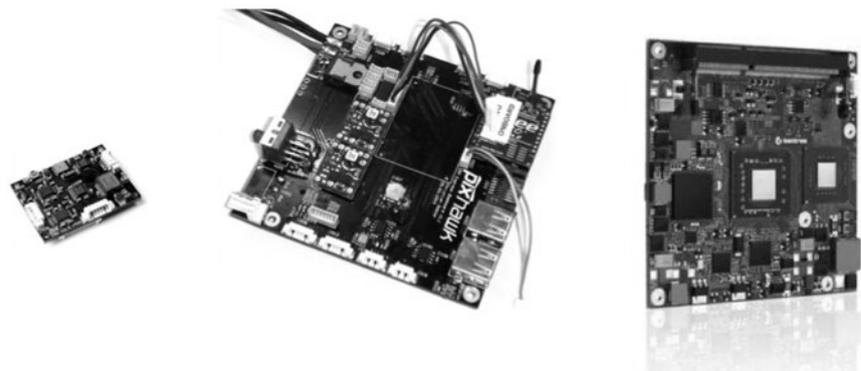


Fig. 8 From left to right: pxIMU Autopilot, pxCOMex image processing module, microETXExpress Core 2 DUO 1.86 GHz module



6.3.1 Autopilot unit

The pxIMU inertial measurement unit/autopilot board (Fig. 8, left part) provides 3D linear acceleration (accelerometer, ± 6 g), 3D angular velocity (± 500 deg/s), 3D magnetic field (\pm milligauss), barometric pressure (130–1030 hectopascal (hPa)) and temperature. The onboard MCU for sensor readout and sensor fusion as well as position and attitude control is a 60 MHz ARM7 microcontroller. It can be flashed via an USB bootloader, and stores settings such as PID parameters in its onboard EEPROM. It provides the required I2C bus to the motor controllers, additional GPIOs, ADC input, and other peripherals. It is interfaced via UART to the computer vision processing unit, and it operates at a maximum update rate of 200–500 Hz.

6.3.2 Image processing unit

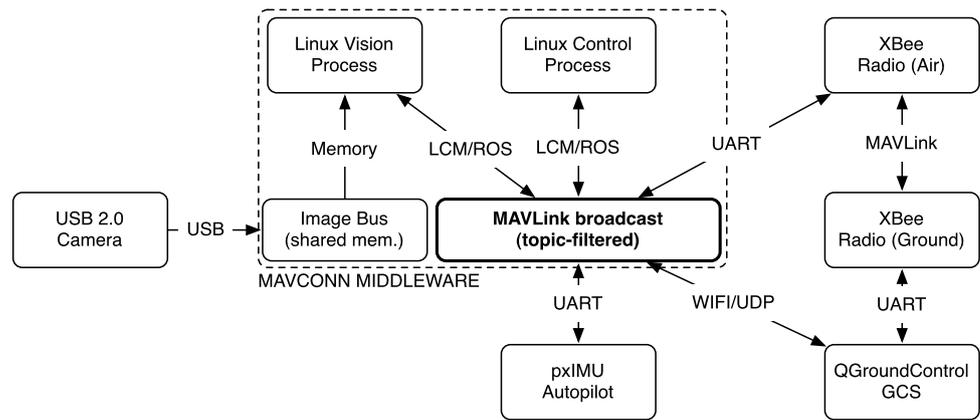
The processing unit is the core piece of the system and consists of a two-board stack. The pxCOMex base board (Fig. 8, middle) provides the USB and UART peripherals to interface machine vision cameras, communication equipment and the pxIMU module. It can accept any micro COM express industry standard module. Currently, a Kontron etX-Express module with Intel@Core™2 Duo 1.86 GHz and 2 GB DDR3 RAM is used (Fig. 8, right), but future upgrade options include Intel@Core™i7 CPUs. It has 4× UART, 7×

USB 2.0 and 1× S-ATA 2.0 peripheral options. The typical onboard setup consists of 4× PointGrey Firefly MV monochrome cameras, 1× USB 2.0 802.11n WiFi adapter, and 1× S-ATA 128 GB SSD with more than 100 MB/s write speed. The pxIMU unit, the GPS module, and the XBee radio modem are connected via UART to the processing unit. With a weight of 230 g including cooling and only 27 W peak power consumption, the processing unit can be easily lifted by a wide range of aerial systems, and not limited to the quadrotor presented here.

6.4 Aerial middleware and message format

Existing middleware solutions for ground robotics include ROS (Quigley et al. 2009), CARMEN (Montemerlo et al. 2003) and CLARAty (Volpe et al. 2001). CARMEN and CLARAty have paved the way for standardized robotics toolkits, but their use has declined with the wide adoption of ROS. Although ROS has been used on MAVs, all of these toolkits assume an Ethernet network to all connected nodes. However, MAV onboard-networks typically include no Ethernet device, but several devices connected via serial links and USB. As these toolkits do not scale down to this link type, every packet has to be transcoded by bridge processes. Therefore, we propose a new communication protocol and architecture that can be transparently used on different hardware links and which minimizes the system complexity.

Fig. 9 MAVCONN Network showing different physical links using the MAVLink protocol



As shown in Fig. 9, the PIXHAWK Linux middleware consists of several layered software components. This architecture allows us to use the different base communication layers (ROS and LCM), and provides a convenient high-level programming interface (API) to the image distribution server. MAVLink messages from the IMU and the ground control station can also be directly received by any process. We rely on the Lightweight Communication Marshalling library (LCM) as the base middleware, as it was shown in Huang et al. (2010) that LCM outperforms ROS in low-latency applications. Another benefit is the increased robustness of the overall software architecture when using LCM, as no central server exists and our communication over MAVLink is mostly stateless. This eliminates a single point of failure (the ROS central node), and also eliminates possible protocol lockups in stateful implementations (as in many ROS nodes). Our system can however still benefit from ROS software packages, such as the ROS Kinect interface, by using our ROS-MAVCONN bridge process that routes between the two software packages.

The mission and control architecture of the presented robotics toolkit is based on a lightweight protocol called MAVLink, which scales from serial to UDP links. It serves also as a communication protocol between the flight computer (pxIMU) and the onboard main computer (pxCOMex/pxOvero). As MAVLink is used on all communication links including the downlink to the operator control unit, it is particularly important that this protocol scales down to very low bandwidth links, and allows the use of several links in parallel. In turn, this parallel use allows several redundant links, which in our case, are long-range XBee radio modems and 802.11n Wifi (UDP). MAVLink has a small 8-byte overhead per packet, allows routing on an inter-system or intra-system level, and has in-built packet-drop detection. Due to the low overhead, it is both suitable for UDP and UART/radio modem transport layers. The efficient encoding also allows protocol execution on microcontrollers. These properties allow the building of a homogeneous communication architecture across the PIXHAWK system. The MAVLink sentences

are generated based on an XML protocol specification file in the MAVLink format. The code generator ensures well-formed messages, and generates C89-compatible C-code for the message packing and unpacking. This allows fast and safe extensions, and changes to the communication protocol, and ensures that no implementation errors will occur for new messages. Our current implementation supports the use of the Lightweight Communication Marshalling Library (LCM) and the Robot Operating System (ROS) as transport layers.

While we use MAVLink to send system states and control commands, we do rely on a separate shared memory implementation of an image hub. This component allows sharing of images of all cameras with an arbitrary number of Linux processes and with the least overhead possible.

6.5 Mission management

A core part of the autonomous flight is the onboard mission management logic, which allows the system to autonomously follow a flight plan or to perform simple tasks, such as sweeping a region of interest. The user can specify these missions in the open-source QGroundControl operator control unit. It is a C++ application using the Nokia Qt toolkit. Communication with the MAV is based on the MAVLink protocol and transported either via UART/radio modem or via WiFi/UDP. QGroundControl covers the whole operational spectrum of an autonomous MAV; it can graph and log system data in real-time, and it provides 2D and 3D moving maps for the flight operation.

Figure 10 shows a typical operator setup: a 3D moving map displaying the waypoint locations, the safety/home location, and the MAV trail (red ellipsoid, simulated for visualization). By interfacing to Google Earth, up-to-date imagery data and 3D models of buildings are available to QGroundControl. The bottom part of the window shows the waypoint list. The operator can either edit the waypoint list or drag the waypoint icons in the 3D interface. The instrument on the top right displays the MAV list with one system

Fig. 10 QGroundControl displaying the current waypoints and trajectory of the system



being active. The circular instrument below shows the controller state (no information available, thus crossed out) and nearby obstacles (no close obstacles present). The gauge instruments can be easily configured by the user to visualize a subset of sensor values received by the GCS.

7 Experiments and results

We conduct experiments to evaluate the image transmission delays in the MAV system, the visual localization without and with the vision-IMU 2-point algorithm, and the stereo obstacle detection, and discuss the results.

7.1 Image transmission delays

Computer vision algorithms can exploit synchronized attitude and vision data to increase accuracy and robustness. Since machine vision cameras have a delay in the tens of milliseconds range due to USB/Firewire transfer time and operating system scheduling delays, the best solution is to synchronize the camera to the inertial measurement unit with a hardware shutter. Figure 11 shows the USB transfer delays (red, bottom curve), the USB and shared memory interface delay (green, middle curve), and the total camera shutter to control output delay (blue, top). The measurements show that the overall delay is in the same range as the

interval between two captured images (36 ms for the presented localization). This applies to a system with low and high CPU loads (both cores at maximum load).

7.2 Visual localization

We perform two experiments to determine the localization accuracy of our ARToolkit+ localization without and with the vision-IMU 2-point algorithm described in Sect. 3.2. In our experiments, we use ground truth data from a Vicon motion tracking system; the ground truth data is provided at a rate of 50 Hz and is very precise with <1 mm error. The objective of the experiments is two-fold: to quantitatively measure the ARToolkit+ localization error relative to the Vicon groundtruth, and to examine whether the 2-point algorithm improves the localization accuracy by using vision-IMU fusion. To be able to localize the helicopter with the described vision system during the whole flight, ARToolkit+ markers were laid out on the floor in the flight area as shown in Fig. 4.

In each experiment, we use our operator control software, QGroundControl, as shown in Fig. 12 to set relevant parameters for MAV software components, monitor the MAV's status, send commands to the MAV, and preset waypoints for autonomous flight.

Furthermore, in each experiment, the MAV executes an autonomous flight; at the beginning and end of the flight, open-loop takeoff and landing are performed respectively,

Fig. 11 Measured system delays. *Top*: Total round-trip time from the moment the camera is triggered by the IMU until the moment the image is transferred to the onboard computer, vision is computed, state estimation is run and the motor controllers are set to the updated value. *Middle*: Trigger to vision process delay. *Bottom*: USB transfer delay. The spikes in the processing time are mostly attributed to the scheduling of the computer vision processes, as these results were obtained on a non-RT Linux system

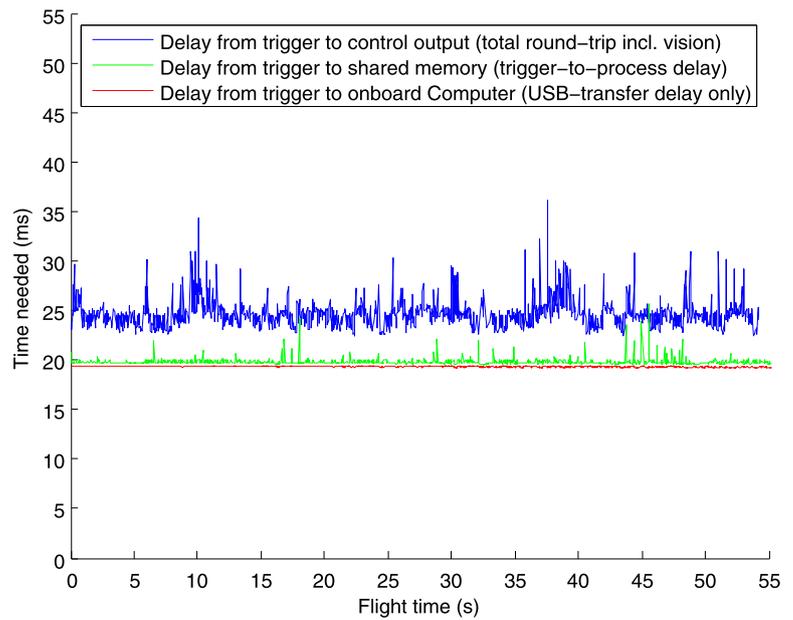
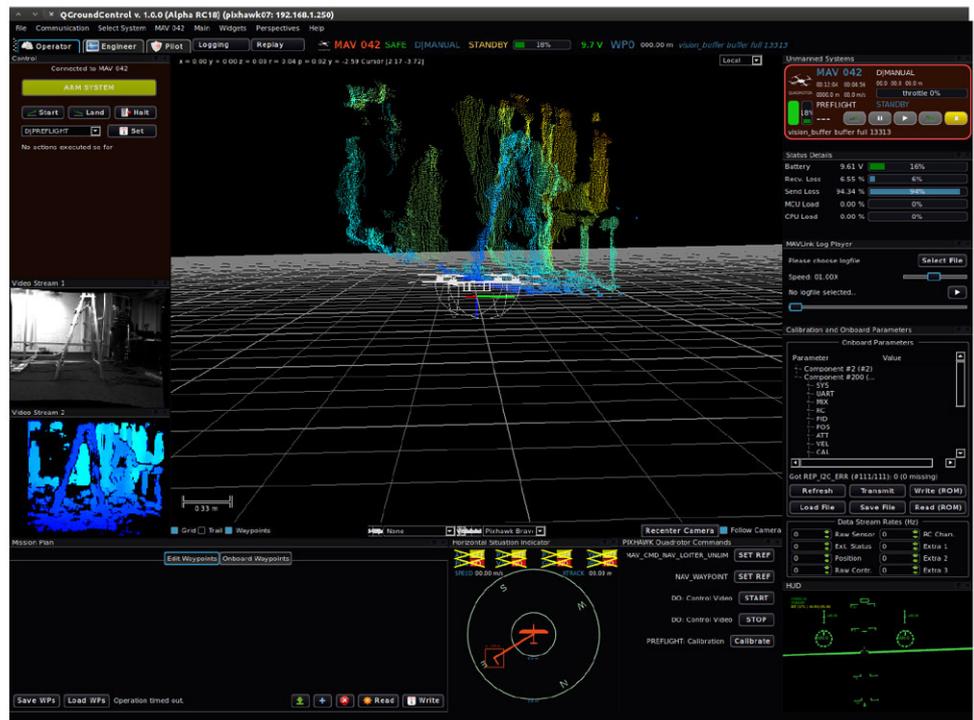


Fig. 12 QGroundControl view with live image streaming from the helicopter using MAVLink over UDP. The live view on the left shows the rectified and depth images from the stereo camera setup



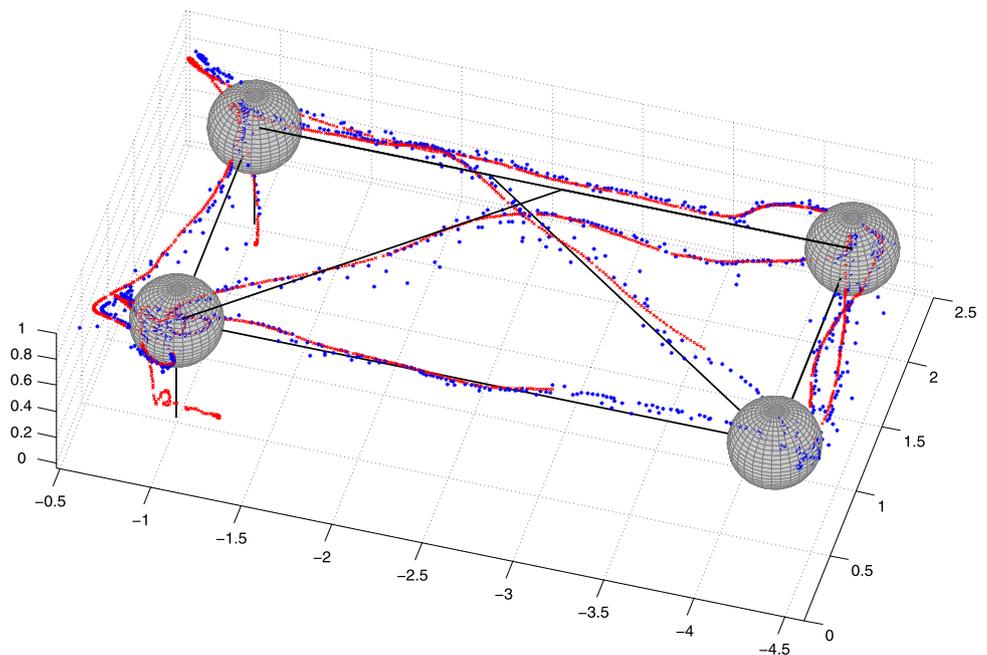
using in the control loop only the estimated state of the MAV without any external position or attitude reference. During the flight, the MAV uses the localization output to follow the preset waypoints.

7.2.1 Experiment 1—autonomous waypoint following

In the first experiment, Fig. 13 shows the localization results using the ARToolkit+ localization without the 2-point algo-

rithm. The plot shows a flight around a rectangular path and two crossings. The solid black line shows the planned flight path; the vertical line in the top left corner of the figure indicates takeoff while the vertical line in the bottom left corner indicates landing. The grey spheres indicate the waypoints; the radius of each sphere equals the acceptance radius within which the waypoint is marked as reached. The blue asterisks represent the position estimates computed by the unfiltered

Fig. 13 Trajectory of an autonomous flight using unfiltered ARToolkit+ localization (*asterisks*) including takeoff and landing plotted together with the Vicon groundtruth (*crosses*) and planned path (*solid line* and spheres are the planned waypoints)



visual localization, and the red crosses represent the Vicon ground truth.

It is observed in Fig. 13 that the ARToolkit+ localization output without the 2-point algorithm closely follows the Vicon ground truth, but is subject to frequent large errors. This is because the localization output is computed purely based on vision, hence making it extremely sensitive to errors from the extracted image features. A small error in the position of the extracted image feature would translate into a large error in the localization output, thus explaining the frequent large errors.

7.2.2 Experiment 2—comparison of IMU-aided localization

The quadrotor autonomously flies a similar trajectory as in Fig. 13. The flight control is based on the ARToolkit localization without using the IMU, and the state estimation is done with an attitude observer filter and four independent 1D Kalman filters for x , y , z and yaw. In our vision-IMU 2-point algorithm, we first compute the 2D image features that correspond to the 4 corners of each ARTK marker in full view in the image; example 2D image features are shown as black circles in Fig. 14. We then establish 2D-3D correspondences through identification of the marker IDs and retrieval of the 3D coordinates of the identified markers from the ARToolkit+ configuration file. We use the same set of 2D-3D correspondences to compute the pose estimates for the ARToolkit+ localization without and with the 2-point algorithm.



Fig. 14 An example of image features (*black circles*) extracted from the 4 corners of ARToolkit+ markers

Figure 15 shows a comparison of the localization output from the ARToolkit+ localization without and with our 2-point algorithm as shown in red and blue respectively; the Vicon readings are shown as groundtruth in green. It is observed from Fig. 15 that the localization output with our 2-point algorithm is significantly smoother and more accurate than that without the 2-point algorithm; the 2-point localization output coincides more closely to the Vicon ground truth and is not subject to large jumps which occur for the localization without the 2-point algorithm. This is due to the additional roll and pitch information from the IMU which helps to reduce the sensitivity of the localization process to errors arising from the extracted image features.

Fig. 15 (Color online) Position estimates from ARToolKit+ localization without (in red) and with (in blue) the 2-point algorithm. The Vicon groundtruth is shown in green

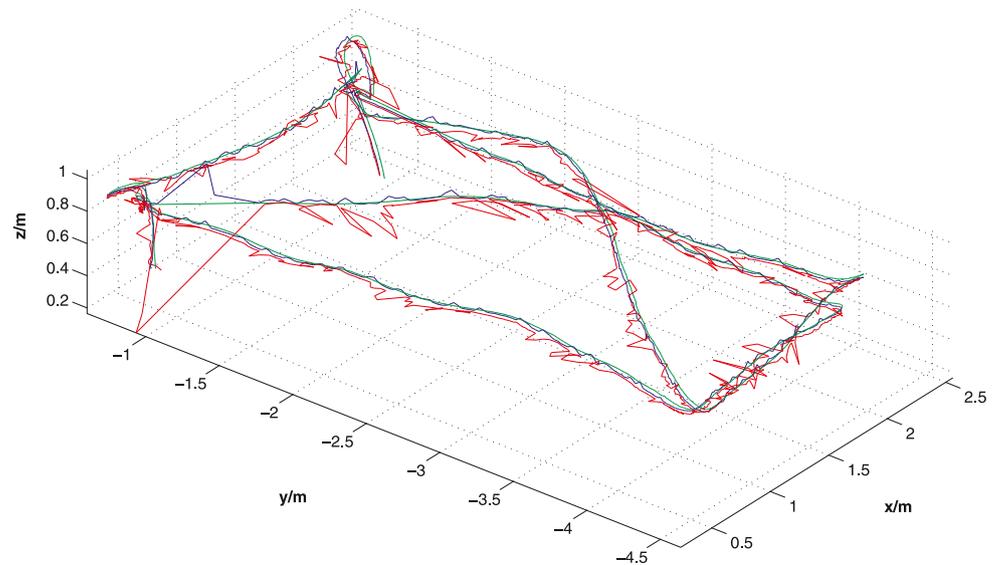


Fig. 16 *Left*: Left stereo image. *Middle*: Colorized point cloud. *Right*: Same point cloud colored by distance from the MAV



In both experiments, the helicopter hovered shortly above the landing position until it reached a steady hovering state, and then landed. It can be observed that there are significant cross-track errors between the actual and planned flight paths. As our focus is not on precise path following, our MAV system is equipped with basic PID position and attitude controllers which are not optimally tuned. Furthermore, the localization output does not reflect the actual position of the MAV, and therefore, deviations from the planned flight path are expected.

7.3 Stereo obstacle detection

We carry out an experiment in which the MAV flies autonomously along preset waypoints. We show a visualization of the stereo processing at one point of time; Fig. 16 shows an image from the left camera of the stereo rig, the resulting 3D point cloud computed from the corresponding stereo frame, and the same point cloud colored by distance from the MAV. In the latter two images, the MAV is shown in green. Figure 17 shows the occupancy map that corresponds to the stereo frame depicted in Fig. 16.

The MAV publishes alert messages if it detects obstacles within a safety clearance of 0.75 m. To test this functionality, we put obstacles (plant, person, cardboard) along one side of

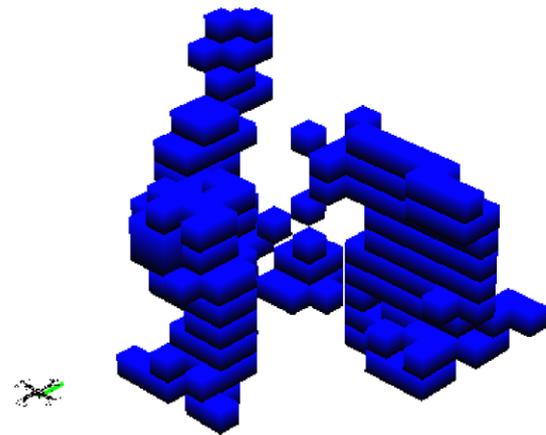


Fig. 17 The occupancy map corresponding to the frame in Fig. 16. Obstacle cells in the map are marked as boxes

the flight path and closer than the pre-set safety clearance. Figure 18 shows the outcome of the test flight; the flight trajectory is shown in blue, while the locations where the MAV published alert messages are marked with red circles. These alert messages could be used by a planning algorithm to change the flight plan.

Fig. 18 Autonomous test flight with obstacles. Circles mark the locations where the MAV published alert messages (when obstacles are detected within the MAV's safety clearance of 0.75 m). The flight trajectory is shown as *solid line*

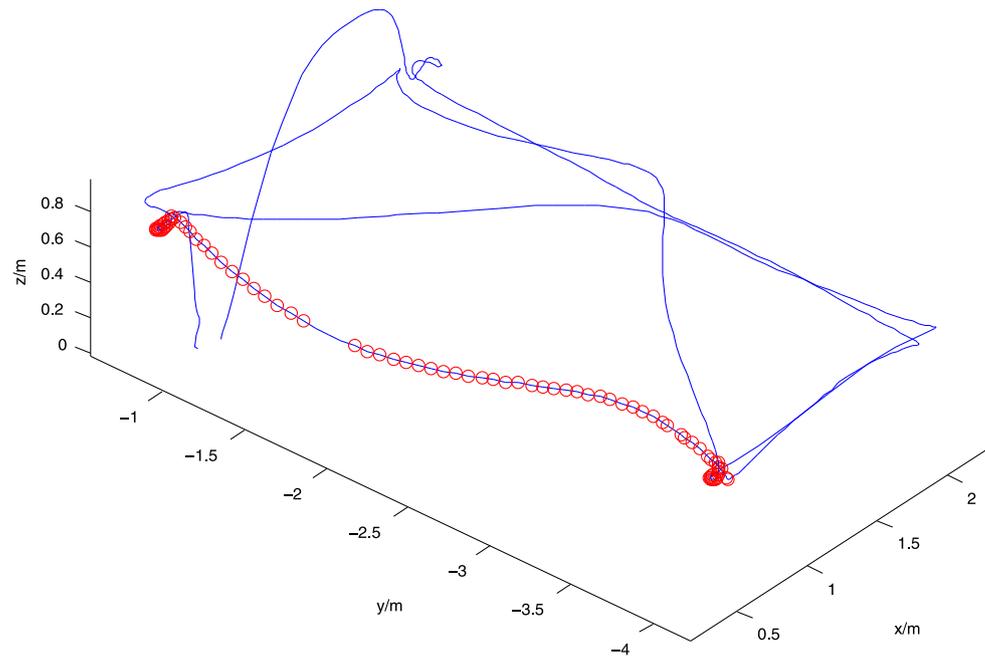


Table 3 Breakdown of computational time for on-board stereo processing using 640×480 stereo images (1.86 GHz Intel@Core™2 Duo)

Process	Average computational time
Image rectification	5 ms
Disparity mapping	29 ms
Point cloud generation	1 ms
Occupancy mapping	50 ms
Total	85 ms

The breakdown of computational time for the stereo processing on-board the MAV is described in Table 3.

8 Conclusions and future works

The PIXHAWK system is a flexible and computationally strong research platform for autonomous micro air vehicles. This system design, especially with a fast onboard computer, is currently unmatched in the class of small-scale MAVs. The hardware IMU-vision synchronization and precise timestamping allows the fusion of IMU and vision information without the need to estimate and take assumptions about the delays. Our results show that fusing vision and IMU information in the proposed way can improve the accuracy of the camera pose estimation, and thus, the overall flight performance. Our platform provides a basic setup for autonomous flight using ARTK+ markers. At the same

time, the system can interface with the stereo cameras, and provide a depth map for obstacle detection.

Our overall system design has proved useful as a research platform, and is intensively used in our group and in several other international research labs.

In future work, we will make use of the powerful on-board computer to do computationally-intensive visual localization with natural features, and autonomous exploration and mapping. Our communication architecture, in particular the MAVLink protocol, supports MAV-to-MAV communication. In addition, we want to exploit this architecture in the direction of distributed localization and distributed mapping of swarms of MAVs.

Acknowledgements We would like to thank our students (in alphabetical order) Bastian Bücheler, Andi Cortinovis, Christian Dobler, Dominik Honegger, Fabian Landau, Laurens Mackay, Tobias Nägeli, Philippe Petit, Martin Rutschmann, Amirehsan Sarabadani, Christian Schluchter and Oliver Scheuss for their contributions to the current system and the students of the previous semesters for the foundations they provided. Raffaello d'Andrea and Sergei Lupashin (ETH IDSC) provided valuable feedback.

This work was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #231855 (sFly) and by the Swiss National Science Foundation (SNF) under grant # 200021-125017.

References

- Achtelik, M., Achtelik, M., Weiss, S., & Siegwart, R. (2011). On-board IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 3056–3063).

- Bachrach, A., de Winter, A., He, R., Hemann, G., Prentice, S., & Roy, N. (2010). Range—robust autonomous navigation in GPS-denied environments. In *Robotics and automation (ICRA), 2010 IEEE international conference on* (pp. 1096–1097). doi:10.1109/ROBOT.2010.5509990.
- Bills, C., Chen, J., & Saxena, A. (2011). Autonomous MAV flight in indoor environments using single image perspective cues. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 5776–5783).
- Blösch, M., Weiss, S., Scaramuzza, D., & Siegwart, R. (2010). Vision based MAV navigation in unknown and unstructured environments. In *Robotics and automation (ICRA), 2010 IEEE international conference on* (pp. 21–28). doi:10.1109/ROBOT.2010.5509920.
- Bosch, S., Lacroix, S., & Caballero, F. (2006). Autonomous detection of safe landing areas for an UAV from monocular images. In *Intelligent robots and systems, 2006 IEEE/RSJ international conference on* (pp. 5522–5527). doi:10.1109/IROS.2006.282188.
- Bouabdallah, S., Murrieri, P., & Siegwart, R. (2004). Design and control of an indoor micro quadrotor. In *Proceedings of int. conf. on robotics and automation*. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.842>.
- Bouabdallah, S., & Siegwart, R. (2007). Full control of a quadrotor. In *Intelligent robots and systems, 2007. IROS 2007. IEEE/RSJ international conference on* (pp. 153–158). doi:10.1109/IROS.2007.4399042.
- Conte, G., & Doherty, P. (2008). An integrated UAV navigation system based on aerial image matching. In *Proceedings of the IEEE aerospace conference*. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.3963&rep=rep1&type=pdf>.
- Dryanovski, I., Morris, W., & Xiao, J. (2011). An open-source pose estimation system for micro-air vehicles. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 4449–4454).
- Ducard, G., & D'Andrea, R. (2009). Autonomous quadrotor flight using a vision system and accommodating frames misalignment. In *Industrial embedded systems, 2009. SIES '09. IEEE international symposium on* (pp. 261–264). doi:10.1109/SIES.2009.5196224.
- Eberli, D., Scaramuzza, D., Weiss, S., & Siegwart, R. (2011). Vision based position control for MAVs using one single circular landmark. *Journal of Intelligent and Robotic Systems*, 61(1–4), 495–512.
- Fowers, S., Lee, D. J., Tippetts, B., Lillywhite, K., Dennis, A., & Archibald, J. (2007). Vision aided stabilization and the development of a quad-rotor micro UAV. In *International symposium on computational intelligence in robotics and automation, 2007. CIRA 2007* (pp. 143–148). doi:10.1109/CIRA.2007.382886.
- Heng, L., Meier, L., Tanskanen, P., Fraundorfer, F., & Pollefeys, M. (2011). Autonomous obstacle avoidance and maneuvering on a vision-guided MAV using on-board processing. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 2472–2477).
- Hofmann, G., Rajnarqan, D., & Waslander, S. (2004). The Stanford testbed of autonomous rotorcraft for multi agent control (starmac). In *Proceedings of digital avionics systems conference (DASC04)*. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1390847.
- Hrabar, S., & Sukhatme, G. (2009). Vision-based navigation through urban canyons. *Journal of Field Robotics*, 26(5), 431–452. doi:10.1002/rob.20284.
- Huang, A., Olson, E., & Moore, D. (2010). LCM: Lightweight Communications and Marshalling. In *Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on* (pp. 4057–4062).
- Johnson, A., Montgomery, J., & Matthies, L. (2005). Vision guided landing of an autonomous helicopter in hazardous terrain. In *Robotics and automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE international conference on* (pp. 3966–3971). doi:10.1109/ROBOT.2005.1570727.
- Kanade, T., Amidi, O., & Ke, Q. (2004). Real-time and 3d vision for autonomous small and micro air vehicles. In *Decision and control, 2004. CDC. 43rd IEEE conference on* (vol. 2, pp. 1655–1662). doi:10.1109/CDC.2004.1430282.
- Kemp, C. (2006). *Visual control of a miniature quad-rotor helicopter*. Ph.D. thesis, Churchill College, University of Cambridge.
- Kukelova, Z., Bujnak, M., & Pajdla, T. (2010). Closed-form solutions to the minimal absolute pose problems with known vertical direction. In *ACCV*.
- Kukelova, Z., Bujnak, M., & Pajdla, T. (2011). Closed-form solutions to minimal absolute pose problems with known vertical direction. In *Computer vision—ACCV 2010*. URL <http://www.springerlink.com/index/M012M78244081306.pdf>.
- Lobo, J., & Dias, J. (2007). Relative pose calibration between visual and inertial sensors. *International Journal of Robotics Research*, 26(6), 561–575.
- Meier, L., Tanskanen, P., Fraundorfer, F., & Pollefeys, M. (2011). Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 2992–2997).
- Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)*.
- Mellinger, D., Shomin, M., Michael, N., & Kumar, V. (2010). Cooperative grasping and transport using multiple quadrotors. In *Proceedings of the international symposium on distributed autonomous robotic systems*.
- Montemerlo, M., Roy, N., & Thrun, S. (2003). Perspectives on standardization in mobile robot programming: the Carnegie Mellon navigation (Carmen) toolkit. In *Intelligent robots and systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ international conference on* (vol. 3, pp. 2436–2441). doi:10.1109/IROS.2003.1249235.
- Proctor, A. A., Johnson, E. N., & Apker, T. B. (2006). Vision-only control and guidance for aircraft. *Journal of Field Robotics*, 23(10), 863–890. doi:10.1002/rob.20155.
- Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). *Ros: An open-source robot operating system*.
- Roy, N., He, R., Bachrach, A., & Achtelik, M. (2009). On the design and use of a micro air vehicle to track and avoid adversaries. *International Journal of Robotics Research*. URL <http://ijr.sagepub.com/cgi/content/abstract/29/5/529>.
- Saripalli, S., Montgomery, J., & Sukhatme, G. (2002). Vision-based autonomous landing of an unmanned aerial vehicle. In *Robotics and automation, 2002. Proceedings. ICRA '02. IEEE international conference on* (vol. 3, pp. 2799–2804). doi:10.1109/ROBOT.2002.1013656.
- Scherer, S., Singh, S., Chamberlain, L., & Elgersma, M. (2008). Flying fast and low among obstacles: Methodology and experiments. *The International Journal of Robotics Research*, 27(5), 549–574. doi:10.1177/0278364908090949.
- Shen, S., Michael, N., & Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 20–25).
- Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R., & Das, H. (2001). The CLARAty architecture for robotic autonomy. In *Aerospace conference, 2001, IEEE proceedings* (vol. 1, pp. 1/121–1/132). doi:10.1109/AERO.2001.931701.

- Wagner, D., & Schmalstieg, D. (2007). Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th computer vision winter workshop*. URL <http://www.icg.tu-graz.ac.at/Members/daniel/ARToolKitPlusMobilePoseTracking>.
- Li, W., Zhang, T., & Khlhnlz, K. (2011). A vision-guided autonomous quadrotor in an air-ground multi-robot system. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 2980–2985).
- Wenzel, K., Masselli, A., & Zell, A. (2011). Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. *Journal of Intelligent Robotic Systems*, 61, 221–238. doi:10.1007/s10846-010-9473-0.
- Williams, B., Hudson, N., Tweddle, B., Brockers, R., & Matthies, L. (2011). Feature and pose constrained visual aided inertial navigation for computationally constrained aerial vehicles. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 431–438).



Lorenz Meier is a Ph.D. student at the Computer Vision and Geometry Lab at ETH Zurich, Switzerland. He received his Master's degree in computer science from ETH Zurich in 2011. His research focus is on real-time and distributed localization for autonomous flight control.



Petri Tanskanen is a Ph.D. student at the Computer Vision and Geometry Lab at ETH Zurich, Switzerland. His research focus is on real-time localization and mapping using IMU-vision techniques on micro air vehicles.



Lionel Heng is a second-year Ph.D. student in the Computer Vision and Geometry Lab at ETH Zürich. His current research focuses on real-time 3D mapping and path planning for vision-guided micro aerial vehicles in dynamic and unknown environments. He received his undergraduate degree in computer science from Carnegie Mellon University in 2006, and his Master's degree in computer science from Stanford University in 2007. He spent three years at DSO National Laboratories where he worked on perception for unmanned ground vehicles. His Ph.D. studies are currently sponsored by the DSO Postgraduate Scholarship.



Gim Hee Lee is a third year Ph.D. student in the Computer Vision and Geometry Group at ETH Zürich, Switzerland. He works on visual Simultaneous Localization and Mapping (SLAM) for micro-aerial vehicles. Before joining ETH, he was at DSO National Laboratories, Singapore, for a year where he worked on localization for ground robots. He received both his M.Eng. and B.Eng. (1st Class Honours) degrees from the Department of Mechanical Engineering in the National University of Singapore on 2008 and 2005 respectively.



Friedrich Fraundorfer received the Ph.D. degree in computer science from TU Graz, Austria in 2006. He was a postdoctoral researcher at the University of Kentucky in 2006 and at the University of North Carolina at Chapel Hill in 2007 where he worked on large scale image search, which was applied to scalable place recognition, loop detection and global localization. He is now senior researcher at the Computer Vision and Geometry Lab at ETH Zurich, Switzerland, where he works on visual SLAM, large-scale visual mapping and general robot vision.



Marc Pollefeys was born in 1971. He is a full professor in the Dept. of Computer Science of ETH Zurich since 2007 where he is the head of the Institute for Visual Computing and leads the Computer Vision and Geometry lab. He currently also remains associated with the Dept. of Computer Science of the University of North Carolina at Chapel Hill where he started as an assistant professor in 2002 and became an associate professor in 2005. Before this he was a postdoctoral researcher at the KU-Leuven in Belgium, where

he also received his M.S. and Ph.D. degrees in 1994 and 1999, respectively. His main area of research is computer vision. One of his

main research goals is to develop flexible approaches to capture visual representations of real world objects, scenes and events. Dr. Pollefeys has received several prizes for his research, including a Marr prize, an NSF CAREER award, a Packard Fellowship and a European Research Council Starting Grant. He is the author or co-author of more than 100 peer-reviewed publications. He was a program chair for the IEEE Conference on Computer Vision and Pattern Recognition 2009 (CVPR) and was general/program chair of the Third Symposium on 3D Data Processing, Visualization and Transmission. Prof. Pollefeys is on the Editorial Board of the International Journal of Computer Vision, Foundations and Trends in Computer Graphics and Computer Vision and the IPSJ Transactions on Computer Vision and Application and was on the editorial board of IEEE Transactions on Pattern Analysis and Machine Intelligence. He is a senior member of IEEE and a member of ACM.