**Noname manuscript No.**
(will be inserted by the editor)

# 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: date / Accepted: date

**Abstract** We describe a novel quadrotor micro air vehicle (MAV) system that is designed to use computer vision algorithms within the flight control loop. One main contribution is the integration of a powerful onboard computer which is a necessity for running state-of-the-art computer vision algorithms with real-time constraints. The system also features IMU-Vision synchronization by hardware time stamping which allows tight integration of IMU measurements into the computer vision pipeline. We describe visual pose estimation from artificial markers and stereo vision integration for obstacle detection. The control architecture is tested in autonomous flights using onboard computed visual position estimates. In experiments and measurements, we show the benefits of our IMU-Vision synchronization for egomotion estimation, demonstrate autonomous flight and stereo vision obstacle detection.

**Keywords** Micro Aerial Vehicles · Quadrotor · Computer Vision · Stereo vision

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

## 1 INTRODUCTION

We introduce a novel quadrotor MAV design, the Pixhawk MAV, which is specifically designed to be a research platform for computer vision based flight control and autonomous flight using computer vision.

One of our main contributions is the integration of a computing board that is powerful enough to handle all image processing and flight control processes onboard on a small scale quadrotor MAV. 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 having a fully autonomous quadrotor MAV significantly closer. Another major contribution is the hardware IMU-camera synchronization of our system. This allows us to be able 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. In addition, we further advance the state-of-the-art by integrating a vision based obstacle detection system onto the MAV system. The stereo computer vision system produces a high detailed depth map, that gives more detailed information about the obstacle compared to basic sensors as infrared or sonar. The capabilities of the system are furthermore demonstrated by vision only autonomous waypoint based flights. The flight accuracy is compared to Vicon groundtruth.

Finally, all the hardware and software designs are made open-source and are published on our web page [1] with the goal to create an open research platform for the community.

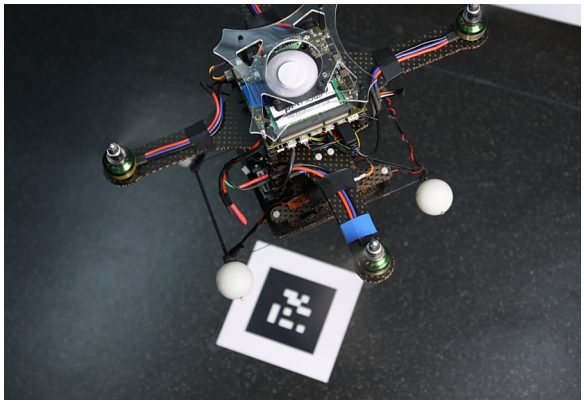

**Fig. 1** PIXHAWK Cheetah Quadrotor

[1] www.pixhawk.ethz.ch

### 1.1 State-of-the-Art

This paper extends our previous work described in [21]. It extends it by a more detailed description and evaluation of the system, i.e. the analysis of the USB image transmission delays. We also add a comparison of a combined IMU-vision pose estimation algorithm 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 (UAV's) has been based on quite large UAV's in the weight range of 10-20kg. UAV's of this size are able to carry an extensive sensor suite, e.g. LIDAR, Radar, camera system and also powerful onboard computers. Impressive results have been shown in terms of autonomous take-off, landing and navigation as well as obstacle avoidance [7,28,16,5,29,25,17,14]. To apply these results to small scale MAV's of under 1.5kg needs specific adaptations to the algorithms and sensor hardware. Recent works successfully demonstrated the use of small LIDAR sensors on such small scale MAV's for mapping and autonomous flight [13,27,8,30,2,11].

Purely vision based autonomous flight control and mapping for small scale MAV's did not yet reach the same level of maturity as with LIDAR sensors.

One of the first works in visual MAV control was done by Kemp [18]. He uses an a-priori generated 3D model of the flight area and 2D-3D edge matching to compute the MAV pose. He demonstrates on-spot hovering of an MAV. The MAV only carries a small analog camera with wireless image transmission. All the processing is done off-board.

More recently, Blösch et al. [4] describe visual autonomous flight using an Asctec Hummingbird and a downward looking camera. A visual SLAM algorithm is running off-board on a standard PC. The images of the on-board camera are streamed to the PC using a USB cable connection (the MAV is actually cable connected, thus limiting the autonomy). Control input is sent back to the MAV via a radio link. In their paper they demonstrate on-spot hovering and waypoint following over a 10m trajectory. To ensure enough visual features for SLAM their testbed is covered with textured posters. The off-board visual SLAM computes positions at varying frame rates between 15-30Hz.

An extension of this work is described by Achtelik et al. [1]. They replace the Asctec Hummingbird with a bigger model, the Asctec Pelican, and equip it with an Intel Atom onboard computer. This allows them to run a modified version of the visual SLAM of [4] on-board with a frame rate of 10Hz. In the paper, they demonstrate on-spot hovering in an indoor and outdoor setting.

Williams et al. [35] use line and point features for visual flight control of a MAV. They describe three types of flight patterns; traversing, hovering and ingress. In their experiments, they compute the MAV trajectory offline (from previously captured images) on a desktop PC for the different flight patterns.

An approach for higher level navigation implemented on a Parrot ARDrone is described by Bills et al. [3]. The Parrot ARDrone comes with a forward and downward looking camera and the feature of onboard on-spot hovering. This makes it an easy to use system. 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. control the direction of movement of the MAV from perspective cues they get from images and from classification of the environment. This allows the MAV to follow corridors and even make turns. However, there is no notion of a metric map, and the image processing is completely done off-board on a desktop PC.

The image processing can greatly be simplified with the use of artificial markers. Artificial markers are used by Eberli et al. [10] for hovering, take-off, and landing. They describe the use of one circular marker to compute the position and pose of the MAV. In their experiments, the MAV is connected via USB cable to a ground station, and the image processing is done off-board on a desktop PC.

A different approach by Li et al. [33] shows hovering, take-off, and landing of an Asctec Hummingbird equipped with an Intel Atom onboard computer. They describe the use of an active LED marker pattern. In their approach, the flight control is done on-board. They demonstrate hovering over a marker pattern which is mounted on top of a ground robot; the MAV follows this ground robot.

A similar approach is described by Wenzel et al. [34]. They demonstrate hovering, take-off, and landing of an Asctec Hummingbird using a marker platform mounted on top of a ground robot. The marker pattern is made of IR LED's and the MAV's position is computed from a Wii-mote sensor fitted to the MAV. The Wii-mote sensor performs hardware image processing, and outputs directly the point coordinates of the detected pattern. The final pose computation is then directly done on the MAV's low-level controller.

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 [12, 22, 23, 9]. These works mainly focus on low-level control problems or higher-level tasks assuming given MAV positions and are using off-board control.

The Pixhawk MAV system design itself is an alternative to other commercially available MAV's, like Asctec MAV's [2], MicroKopter [3], MicroDrones [4] or Parrot ArDrone [5]. The hardware design is quite similar to commercially available MAV's. 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

[2] www.asctec.de
[3] www.mikrokopter.de
[4] www.microdrones.com
[5] ardrone.parrot.com

on-board computers. 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)*

| System | PIXHAWK | Asct. Pelican | ARDrone | 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 | - |
| 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 |
| 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 1.1.1 shows the difference between different quadrotor platforms. While the autopilot capabilities of the PIXHAWK quadrotor are similar to another competitive systems, the onboard computational speed, RAM and solid-state disk interfaces are unmatched in the MAV domain.

### *1.1.2 Comparison of MAVCONN middleware*

| Software | MAVCONN | LCM | ROS |
|---|---|---|---|
| Message format | x | - | x |
| Ground Control Station 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 us | 100-1100 us | 500-1100 us |
| Stereo triggering | x | - | - |
| IMU sync | x | - | - |

Table 1.1.2 compares different middlewares. As MAVCONN uses internally LCM as transport layer, all features available in LCM are retained in MAVCONN. 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.

## 2 System Design

The PIXHAWK design includes a powerful onboard computer which makes it possible to run high-level task, 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 Core2Duo. The visual localization module computes the full 6DOF pose of the MAV (see details in section 3). The stereo obstacle detection module computes real-time stereo from the front looking stereo pair (see section 4). The output of the stereo module can be used for obstacle avoidance by the planning module. The planning module currently implements waypoint following. Attitude and position controller are implemented on a low-level realtime controller (see section 5 for state estimation). The position controller takes as input 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 hardware triggering the cameras by the IMU and timestamping the measurements.

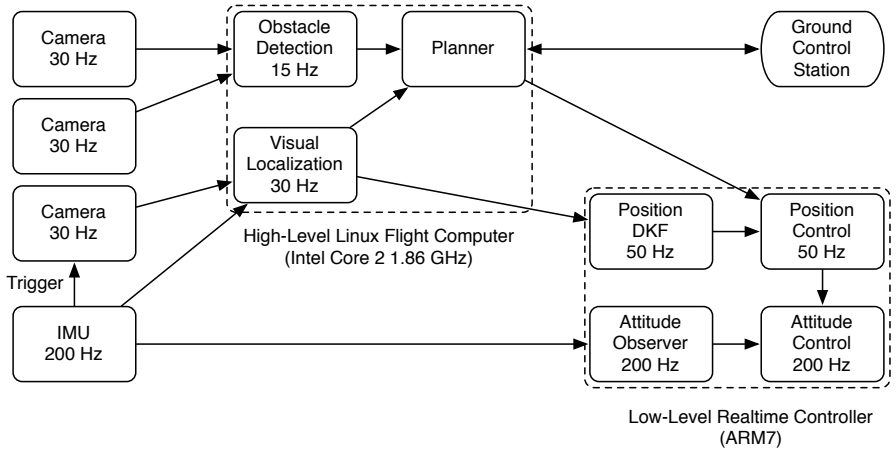


**Fig. 2** The PIXHAWK quadrotor system design. The powerful onboard computer enables high-level tasks as visual localization, obstacle detection and planning run onboard. Position and attitude estimation are implemented on a low-level realtime controller.

### 2.1 Vision-IMU Synchronization

Data from different sensors (in particular from multiple digital cameras) is synchronized by an electronic shutter signal and put together into a timestamped sensor message. IMU sensor data (e.g. absolute attitude and angular

rates) is available as part of the image metadata then. Images are transmitted over USB to the camera process, while the IMU measurements and the shutter timestamp are delivered through a serial interface. Image transmission from camera to main memory via USB takes approximately 16-19 ms (where the time differs sligthly for each image), while the transmission of the shutter timestamp from 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 the full vision-IMU dataset to the software middleware. Fig. 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.

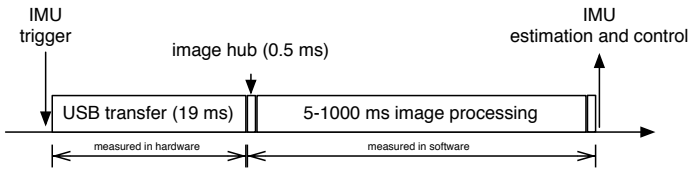


**Fig. 3** System delays. Overall delay varies between 25 and >1000 ms. USB and UART transfer delays are only observable with hardware triggering, else they remain unknown.

## 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. Fig. 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 position is fed back to the position estimation and control software module running on the ARM7 autopilot controller.

### 3.1 ARTK+ Localization

A localization test bed was created that uses markers with an adapted implementation of ARToolkit+ [32] 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 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 presence of 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 they all expect some kind of covariance estimate of the vision position data. This is not possible in a clean way, as vision based localization typically works on 2D image features that are matched between two images. Since this matching is not perfect outlier removal is necessary which again cannot be made error free and the remaining outliers in the feature correspondences will disturb the position estimate. The covariance resulting from a non-linear optimization of the vision algorithm cannot correctly capture these misestimates due to outliers in the feature correspondences. The more promising method is to include the IMU data directly in to the vision estimation to be able to use the information during the outlier filtering and position estimation.

If the accuracy of the IMU estimated vertical direction is higher than the pure vision estimate then it can be included into the vision algorithm to improve the localization accuracy. Fig 15 shows that the localization accuracy

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

increases when using two points and the known vertical (from the inertial data) instead of four points. In case of the 2-point algorithm [19], the calculation steps for the pose estimation are significantly simplified when substituting parts of the calculation 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 as outlier filter for image features, so the benefits of directly combining IMU and vision are, depending on the methods used, improved accuracy and computation time.

Since the roll and pitch angle 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 equation 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 homogenous coordinate space. This ray is then rotated by roll and pitch with the rotation matrix formed by multiplying the rotation matrices around roll and pitch $R_{\phi\theta}$. The resulting ray in homogenous coordinates is now fronto-parallel with respect to the ground plane.

$$u' = R_{\phi\theta} K^{-1} u \tag{1}$$

The resulting fronto-parallel view has only $x$, $y$ and $\psi$ as free parameters. Kukelova et al. provided a closed-form solution to localize from two points and known vertical direction by solving for $x$, $y$, $z$ and $\psi$. By applying this

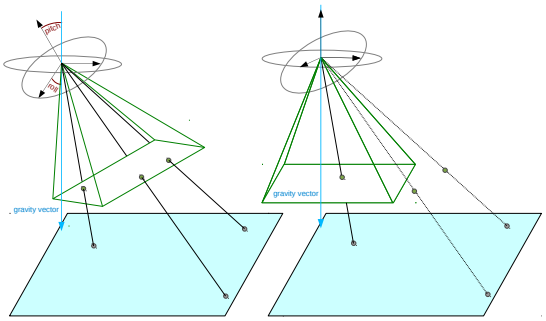


**Fig. 5** Relation of gravity vector and camera angles. The right-fronto parallel view is obtained by rotating the image plane by the roll and pitch angles.

algorithm in a least squares sense on the ARToolkit correspondences, we obtain the final position output, since $\phi$ and $\theta$ were already known. Fig. 15 shows that the solution of the vision-IMU 2-point algorithm outperforms the 4-point algorithm used in ARToolkit+ in accuracy. Both algorithms operated 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 as the global reference. This is especially important for loop closure in SLAM where a global attitude information can faciliate loop detection and reduces convergence into local minimas. This also explains the advantage of storing the absolute attitude as meta data along the images rather than the relative to the last frame. From 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 $\psi$ is filtered with a block 4x1D Kalman filters in the next step, which implies that the filters are parameterized with an error model of the computer vision approach. As IMU and vision both estimate the 3 degree of freedom 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 wrong attitude estimate because of the projective dependency of 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 already into the vision estimation in Fig 15. The variance on the position is reduced by a significant amount such that there is no need for subsequent outlier removal anymore.

## 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. In our stereo processing pipeline, we compute disparity data from stereo image pairs, and subsequently, compute a point cloud which is used for obstacle avoidance. If there is a cluster of points with a minimum density that 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 x 480 disparity map. Subsequently, we compute the depth to the points in the scene relative to the camera coordinate system:

$$z = \frac{bf}{d} \tag{2}$$

where $d$ is the disparity. Differentiation of Equation 2 with respect to $d$ yields:

$$\Delta z = \frac{bf}{d^2}\Delta d \tag{3}$$

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

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

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:

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix} = \frac{z}{f} \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \tag{5}$$

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:

$$\begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \end{bmatrix} = {}^{imu}_{world}H\,{}^{cam}_{imu}H \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix} \tag{6}$$

where ${}^{imu}_{world}H$ is the homogeneous transform from the world frame to the IMU frame and is estimated by the visual odometry, and ${}^{cam}_{imu}H$ is the homogeneous transform from the IMU frame to the camera frame and is estimated using the InerVis calibration toolbox [20].

## 5 State Estimation and Control

The state estimation is run on the low-level controller to satisfy the realtime requirements of the system. In addition this allows to compensate visual localization loss or the loss of the complete onboard computer connection. In such an event the system performs an open-loop safety landing maneuver. To synchronize inertial data and vision estimations, the IMU keeps a buffer of attitude measurements for the last $n$ image frames. Once it 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 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 [6], the dynamics can be modeled as three independent dimensions. As the yaw angle is of much better accuracy and resolution in indoor settings from computer vision than 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 x 1D Kalman filters with position and speed as states. The Kalman filter assumes a constant speed model, and takes position as input. The estimated velocity is critical to damp the system, as the only physical damping is the air resistance on the horizontal plane, which is not significant at the hovering or 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 then again, 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 is assuming 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 timebase, the filter uses the measured inter-frame interval as time difference input $\delta t$. 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 position is 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 finally can 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 smoothened 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. Attitude control was implemented following the standard PID based attitude control approach for quadrotors using one PID controller for roll, pitch and yaw each. 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 liftoff weight. In contrast to systems using local stabilization approaches on specialized microcontroller hardware (Parrot ARDrone), 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. Computer vision in contrast 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 $\Delta t$ and a fixed processing delay $\Delta p$. 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 capture 640 x 480 or 752 x 480 grayscale images at up to 60 Hz. The camera interface is open to support additional camera models. However, since there is no standard for trigger-support used by different camera module manufacturers it is necessary to implement a small interface class in the middleware for every new camera type. The camera pair is rigidly mounted with a baseline of 5cm on a carbon composite frame as shown in Figure 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 400-800g 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 450-600g thrust, enabling the system to lift a 400g 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 150-180W for hovering, while the onboard computer consumes only 27 W peak. Therefore, flight time is governed by the weight of the system.

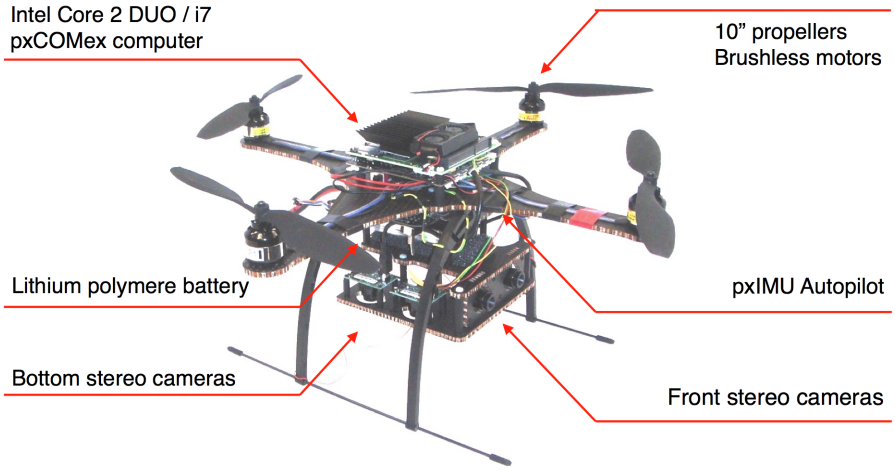


**Fig. 6** Quadrotor Overview

## 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. 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.

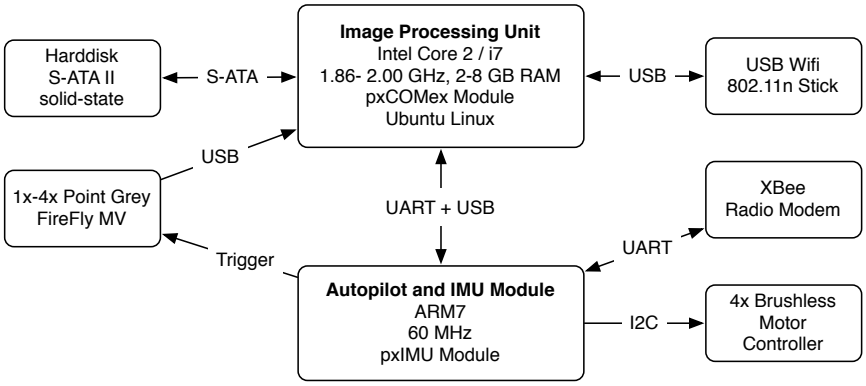


**Fig. 7** Onboard sensors and avionics with electronic buses.

### *6.3.1 Autopilot Unit*

The pxIMU inertial measurement unit/autopilot board (Fig. 8, left part) provides 3D linear acceleration (accelerometer, ±6g), 3D angular velocity (±500 deg/s), 3D magnetic field (± 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 and 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.

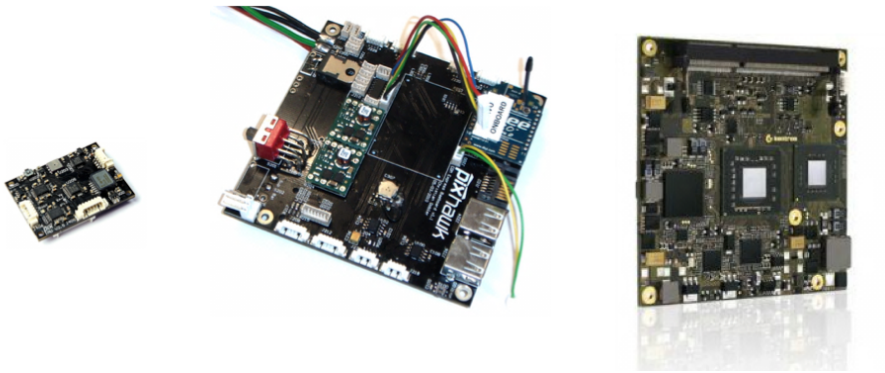

**Fig. 8** From left to right: pxIMU Autopilot, pxCOMex image processing module, microETXExpress Core 2 DUO 1.86 GHz module

*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 etxExpress module with Intel Core 2 DUO 1.86GHz and 2 GB DDR3 RAM is used (Fig. 8, right), but future upgrade options include Intel i7 CPUs. It has 4x UART, 7x USB 2.0 and 1x S-ATA 2.0 peripheral options. The typical onboard setup consists of 4x PointGrey Firefly MV monochrome, 1x USB 2.0 802.11n WiFi adapter and 1x 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 [26], CARMEN [24] and CLARAty [31]. 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 onboard Ethernet networks, 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 minimizes the system complexity.

As shown in Figure 9, the PIXHAWK Linux middleware consists of several layered software components. This architecture allows 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 in any process. We rely on the Lightweight Communication Marshalling library (LCM) as the base middleware, as it was shown in [15] 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.

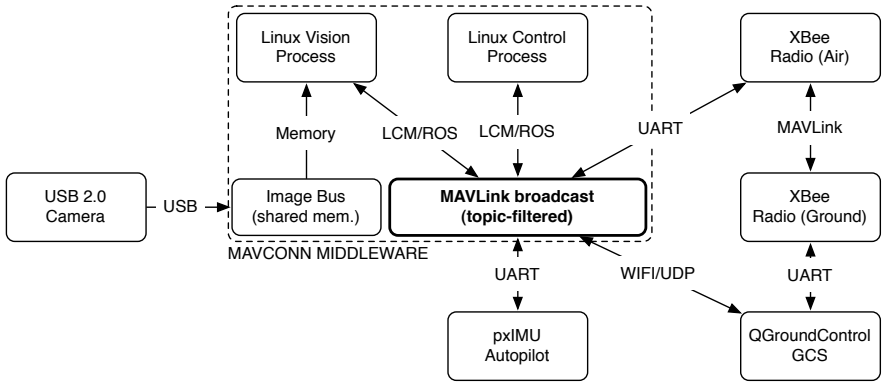


**Fig. 9** MAVCONN Network showing different physical links using the MAVLink protocol.

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 bytes overhead per packet, allows routing on an inter-system or intra-system level, and has an inbuilt 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 allowed the building of a homogenous 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 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 flightplan 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 realtime and it provides 2D and 3D moving maps for the flight operation.


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

Fig. 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 data and building 3D models 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 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 configured freely by the user to any sensor value 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, it is the best solution to synchronize the camera to the inertial measurement unit with a hardware shutter. Fig. 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 leads to a large phase shift of the control output, and therefore, has to be properly measured and compensated. Besides the static delay, the plot also shows that the vision algorithm increases the delay time to between 24 ms and 35 ms; the increase is double its average processing time of about 5 ms.

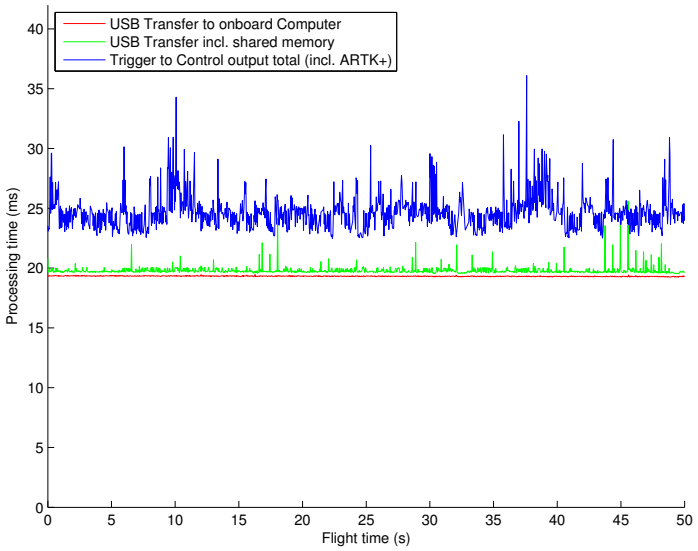


**Fig. 11** Delay of USB and vision processes - note the non-static delay of the overall position output.

### 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 Section 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 $< 1mm$ 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, 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.

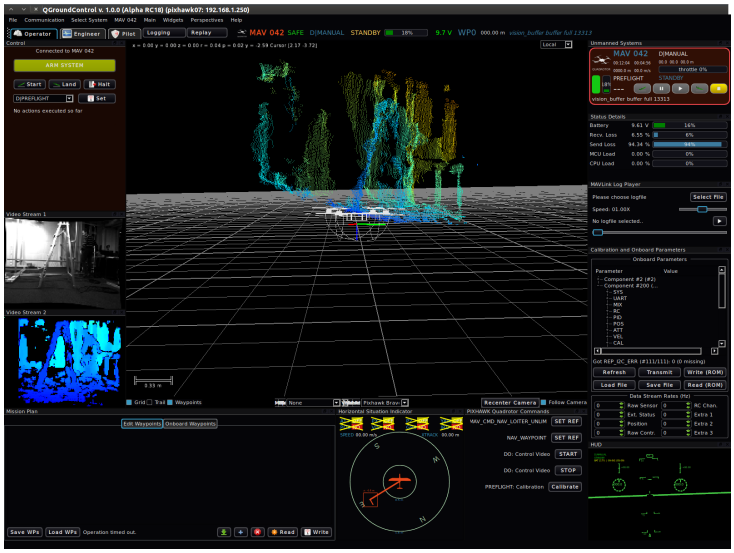

**Fig. 12** QGroundControl view with live image streaming from the helicopter using MAVLink over UDP. The live view on the left shows the rectified and depth images from the stereo camera setup.

### *7.2.1 Experiment 1 - Autonomous waypoint following*

In the first experiment, Fig. 13 shows the localization results using the AR-Toolkit+ localization without the 2-point algorithm. 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 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.

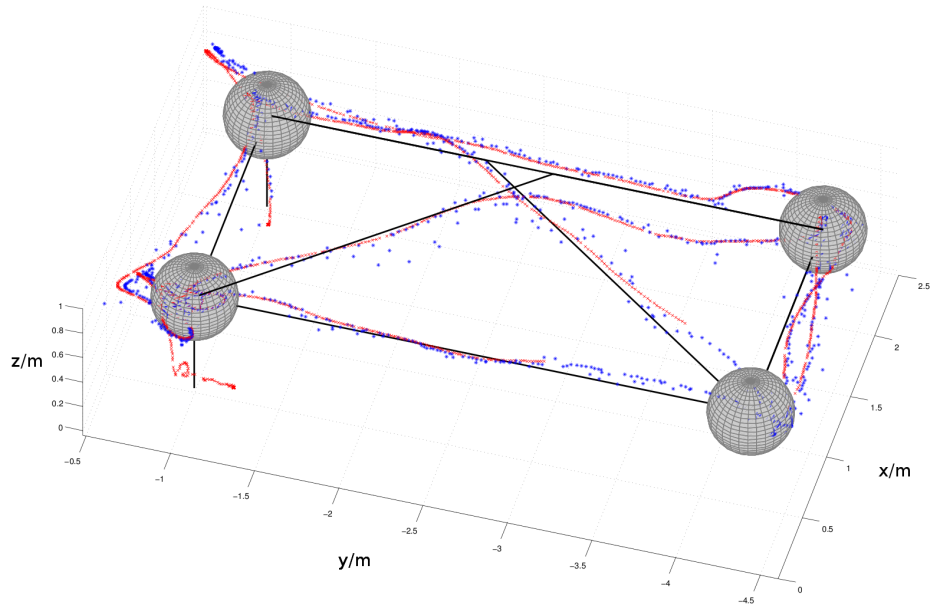


**Fig. 13** Trajectory of an autonomous flight using unfiltered ARToolkit+ localization (blue asterisks) including takeoff and landing plotted together with the Vicon groundtruth (red crosses) and planned path (solid line and spheres are the planned waypoints).

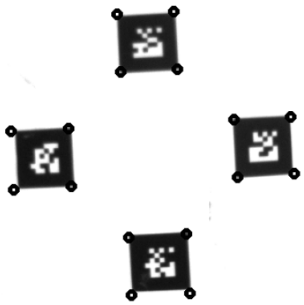

**Fig. 14** An example of image features (black circles) extracted from the 4 corners of ARToolKit+ markers.

### *7.2.2 Experiment 2 - Comparison of IMU-aided localization*

The quadrotor autonomously flies a similar trajectory as in Fig. 13. 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. 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.

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.

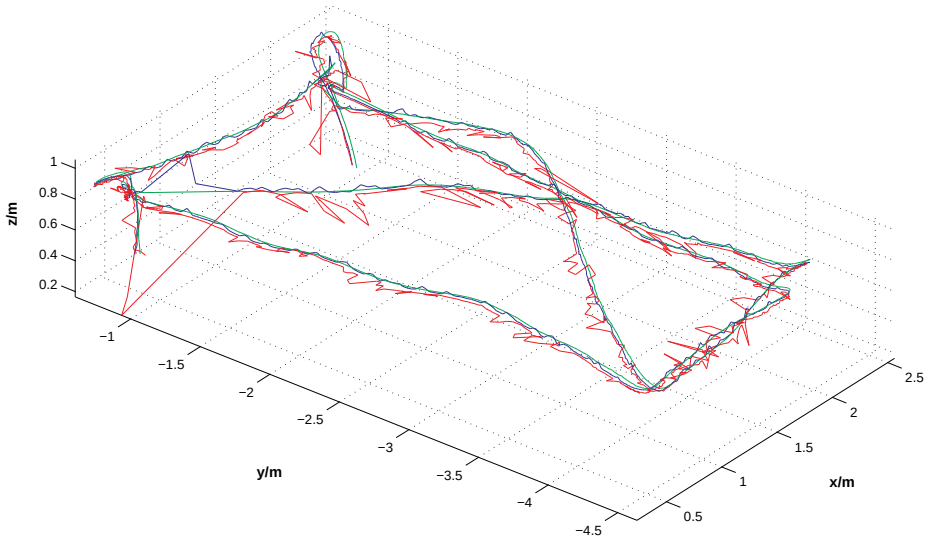


**Fig. 15** Position estimates from ARToolKit+ localization without (in red) and with (in blue) the 2-point algorithm. The Vicon groundtruth is shown in green.

### 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; figure 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.


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

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

**Table 1** Breakdown of computational time for on-board stereo processing using 640 x 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 |
| **Total** | 35 ms |

cardboard) along one side of the flight path and closer than the pre-set safety clearance. Figure 17 shows the outcome of the test flight. The flight trajectory is shown in blue. 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.

The breakdown of computational time for the stereo processing on-board the MAV is described in Table 1.

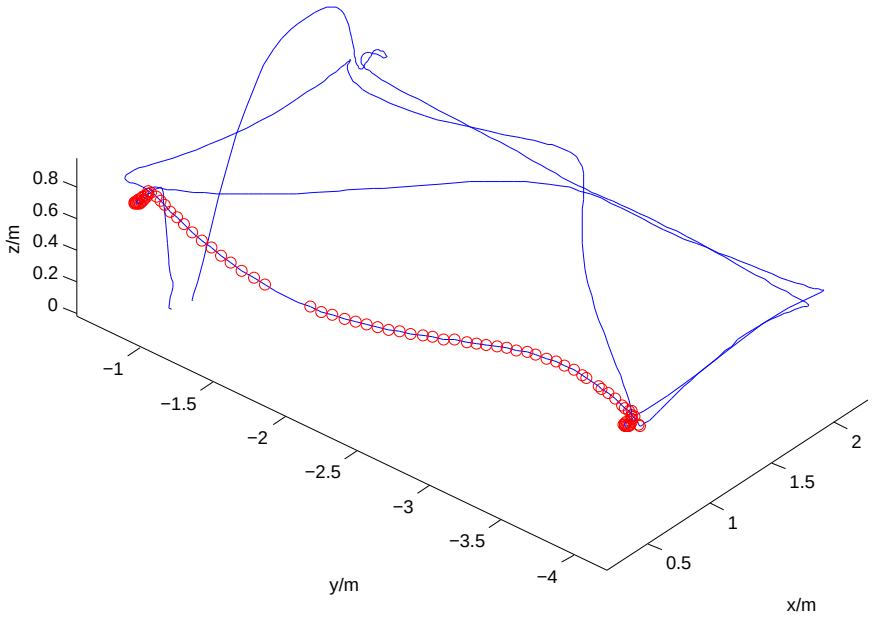


**Fig. 17** Autonomous test flight with obstacles. Red circles mark the locations where the MAV published alert messages (when obstacles are detected within the MAV's safety clearance of 0.75m). The flight trajectory is in blue.

## 8 CONCLUSIONS AND FUTURE WORKS

The PIXHAWK system is a flexible and computationally strong research platform for autonomous micro air vehicles. Especially if a fast onboard computer is needed this system design is currently unmatched in the class of small scale MAV's. 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 the stereo cameras and provide a depth map for obstacle detection.

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

Future work will be to exploit the powerful onboard computer to do computationally more demanding visual localization with natural features and autonomous exploration and mapping. Our communication architecture, in particular the MAVLink protocol, supports MAV-to-MAV communication. We aslo want to exploit this 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, 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 constant and valuable feedback.

## References

1. Achtelik, M., Achtelik, M., Weiss, S., Siegwart, R.: Onboard 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 (2011)
2. Bachrach, A., de Winter, A., He, R., Hemann, G., Prentice, S., Roy, N.: Range - robust autonomous navigation in gps-denied environments. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 1096 –1097 (2010). DOI 10.1109/ROBOT.2010.5509990
3. Bills, C., Chen, J., Saxena, A.: Autonomous mav flight in indoor environments using single image perspective cues. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 5776–5783 (2011)
4. Blösch, M., Weiss, S., Scaramuzza, D., Siegwart, R.: Vision based mav navigation in unknown and unstructured environments. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 21 –28 (2010). DOI 10.1109/ROBOT.2010.5509920
5. Bosch, S., Lacroix, S., Caballero, F.: 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 (2006). DOI 10.1109/IROS.2006.282188
6. Bouabdallah, S., Siegwart, R.: Full control of a quadrotor. In: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pp. 153 –158 (2007). DOI 10.1109/IROS.2007.4399042

7. Conte, G., Doherty, P.: An integrated uav navigation system based on aerial image matching. Proceedings of the IEEE Aerospace Conference (2008). URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.3963&rep=rep1&type=pdf
8. Dryanovski, I., Morris, W., Xiao, J.: An open-source pose estimation system for micro-air vehicles. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 4449–4454 (2011)
9. Ducard, G., D'Andrea, R.: 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 (2009). DOI 10.1109/SIES.2009.5196224
10. Eberli, D., Scaramuzza, D., Weiss, S., Siegwart, R.: Vision based position control for mavs using one single circular landmark. Journal of Intelligent and Robotic Systems **61**(1-4), 495–512 (2011)
11. Fowers, S., Lee, D.J., Tippetts, B., Lillywhite, K., Dennis, A., Archibald, J.: Vision aided stabilization and the development of a quad-rotor micro uav. International Symposium on Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. pp. 143 –148 (2007). DOI 10.1109/CIRA.2007.382886
12. Heng, L., Meier, L., Tanskanen, P., Fraundorfer, F., Pollefeys, M.: 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 (2011)
13. Hofiann, G., Rajnarqan, D., Waslander, S.: The stanford testbed of autonomous rotorcraft for multi agent control (starmac). Proceedings of Digital Avionics Systems Conference (DASC04) URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1390847
14. Hrabar, S., Sukhatme, G.: Vision-based navigation through urban canyons. Journal of Field Robotics **26**(5), 431–452 (2009). DOI 10.1002/rob.20284. URL http://dx.doi.org/10.1002/rob.20284
15. Huang, A., Olson, E., Moore, D.: LCM: Lightweight Communications and Marshalling. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 4057–4062 (2010)
16. Johnson, A., Montgomery, J., Matthies, L.: 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 (2005). DOI 10.1109/ROBOT.2005.1570727
17. Kanade, T., Amidi, O., Ke, Q.: 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 Vol.2 (2004). DOI 10.1109/CDC.2004.1430282
18. Kemp, C.: Visual control of a miniature quad-rotor helicopter. Ph.D. thesis, Churchill College, University of Cambridge (2006)
19. Kukelova, Z., Bujnak, M., Pajdla, T.: Closed-form solutions to minimal absolute pose problems with known vertical direction. Computer Vision–ACCV 2010 (2011). URL http://www.springerlink.com/index/M012M78244081306.pdf
20. Lobo, J., Dias, J.: Relative pose calibration between visual and inertial sensors. International Journal of Robotics Research **26**(6), 561–575 (2007)
21. Meier, L., Tanskanen, P., Fraundorfer, F., Pollefeys, M.: Pixhawk: A system for autonomous flight using onboard computer vision. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 2992–2997 (2011)

22. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2011)
23. Mellinger, D., Shomin, M., Michael, N., Kumar, V.: Cooperative grasping and transport using multiple quadrotors. In: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems (2010)
24. Montemerlo, M., Roy, N., Thrun, S.: 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 vol.3 (2003). DOI 10.1109/IROS.2003.1249235
25. Proctor, A.A., Johnson, E.N., Apker, T.B.: Vision-only control and guidance for aircraft. Journal of Field Robotics **23**(10), 863–890 (2006). DOI 10.1002/rob.20155. URL `http://dx.doi.org/10.1002/rob.20155`
26. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: Ros: An open-source robot operating system (2009)
27. Roy, N., He, R., Bachrach, A., Achtelik, M.: On the design and use of a micro air vehicle to track and avoid adversaries. International Journal of Robotics Research (2010). URL `http://ijr.sagepub.com/cgi/content/abstract/29/5/529`
28. Saripalli, S., Montgomery, J., Sukhatme, G.: 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 (2002). DOI 10.1109/ROBOT.2002.1013656
29. Scherer, S., Singh, S., Chamberlain, L., Elgersma, M.: Flying fast and low among obstacles: Methodology and experiments. The International Journal of Robotics Research **27**(5), 549–574 (2008). DOI 10.1177/0278364908090949. URL `http://ijr.sagepub.com/content/27/5/549.abstract`
30. Shen, S., Michael, N., , Kumar, V.: Autonomous multi-floor indoor navigation with a computationally constrained mav. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 20–25 (2011)
31. Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R., Das, H.: The claraty architecture for robotic autonomy. In: Aerospace Conference, 2001, IEEE Proceedings., vol. 1, pp. 1/121 –1/132 vol.1 (2001). DOI 10.1109/AERO.2001.931701
32. Wagner, D., Schmalstieg, D.: Artoolkitplus for pose tracking on mobile devices. Proceedings of 12th Computer Vision Winter Workshop (2007). URL `http://www.icg.tu-graz.ac.at/Members/daniel/ARToolKitPlusMobilePoseTracking`
33. Wei Li Tianguang Zhang, K.K.: 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 (2011)
34. Wenzel, K., Masselli, A., Zell, A.: Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle. Journal of Intelligent Robotic Systems **61**, 221–238 (2011). URL `http://dx.doi.org/10.1007/s10846-010-9473-0`. 10.1007/s10846-010-9473-0
35. Williams, B., Hudson, N., Tweddle, B., Brockers, R., Matthies, L.: 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 (2011)