2013 International Conference on Unmanned Aircraft Systems (ICUAS)
May 28-31, 2013, Grand Hyatt Atlanta, Atlanta, GA

On-Board Dual-Stereo-Vision for Autonomous Quadrotor Navigation

Konstantin Schauwecker and Andreas Zell

Abstract— **We present a quadrotor Micro Aerial Vehicle (MAV) capable of autonomous indoor navigation. The MAV is equipped with four cameras arranged in two stereo configurations. One camera pair is facing forward and serves as input for a reduced stereo SLAM system. The other camera pair is facing downwards and is used for ground plane detection and tracking. All processing, including sparse stereo matching, is run on-board in real-time and at high processing rates. We demonstrate the capabilities of this MAV design in several flight experiments. Our MAV is able to recover from pose estimation errors and can cope with processing failures for one camera pair. We show that by using two camera pairs instead of one, we are able to significantly increase navigation accuracy and robustness.**

I. Introduction

Micro Aerial Vehicles (MAVs) are becoming increasingly popular in the robotics research community. In particular, much attention has been drawn to quadrotor MAVs, which can now even be obtained by consumers at relatively low costs. Quadrotors are able to take-off and land vertically, hover in one place and move at low speeds. These properties make them ideal for being used inside buildings or other confined spaces. When constructing a robotic MAV, however, achieving autonomous indoor flight is particularly challenging. The key problem is that we generally lack GPS coverage. Hence, other measures are required for indoor-navigation, for which we need additional sensors.

The payload that can be carried by an MAV is, however, considerably low, as is the available on-board power. This greatly limits the choice of available sensors that such an autonomous MAV could be equipped with. In particular, sensors that enable an accurate 3D-localization, such as LIDARS, are usually too heavy to be carried and have a high power consumption. This draws attention to cameras, which only require little energy and can be built very lightly. Cameras can be used for three-dimensional localization within an unknown environment, as has e.g. been shown in [1]. This circumstance has led to significant research on using cameras for the navigation of autonomous MAVs.

Most early approaches relied on using a monocular camera for this purpose. However, the problem with using monocular cameras is that they only allow the estimation of the MAV position with respect to an unknown and unobservable scaling factor. Only very recently it has been possible to use two cameras in a stereo configuration, while performing stereo processing on an on-board computer. If stereo vision is used for MAV navigation, the scaling factor problem disappears and the absolute position can be observed, thanks to additional information from depth measurements.

The cameras are often mounted on the MAV in a forward-facing stereo configuration. This provides a large field of view and can enable the detection of obstacles in flying direction. Forward-facing cameras, however, do not perform well when encountering fast yaw rotations. In this case, a forward-facing camera registers large image movements or might be subject to motion blur. Both effects have a degrading impact on current visual navigation methods. A downward-facing camera, on the other hand, is generally much better at observing yaw rotations and might also – in the case of stereo vision – be used for obtaining measurements for altitude, roll and pitch. However, fast horizontal movements of the MAV will be poorly observable in case of ground proximity flight. Further, stereo matching will not be possible at take-off, as a minimum ground-distance is required.

The dissimilar strengths and weaknesses of forward- and downward-facing cameras lead us to believe that they will complement each other when used in a combined setting. In this paper we present an MAV, which does exactly that: we use both, a forward-facing and a downward-facing camera pair. This means that our MAV is equipped with four cameras in total and we are running stereo matching twice. Even though this greatly increases the amount of data that has to be processed, we are still able to perform all processing on-board and in real-time. We show in this paper that the additional use of a downward-facing camera significantly improves the self-localization, and hence the accuracy of the autonomous flight. To our knowledge, we are the first to realize stereo matching with more than two cameras on-board of an MAV.

II. Related Work

The key to autonomous flight is to enable the MAV to estimate its current pose (i.e. position and orientation) with respect to its environment. While such an estimate can be obtained by integrating readings from an Inertial Measurement Unit (IMU), the accuracy of this estimate quickly degrades with integration time. Visual motion estimation, known as visual odometry, can serve as a more accurate source for pose inference. As we have already mentioned, however, the MAV position can only be observed with respect to an unknown scaling factor, if only one camera is used. This is why most MAVs featuring monocular vision are only operable in specific environments.

One example is the quadrotor MAV presented in [2] that relies on visual markers. This MAV, however, does not perform any on-board image processing, but wirelessly transmits

K. Schauwecker and A. Zell are with the Chair of Cognitive Systems, Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany

978-1-4799-0817-2/13/$31.00 ©2013 IEEE

the camera images to a ground computer. The computer then remotely controls the MAV. An example for an approach with on-board processing can be found in [3], where the authors track the location of a landing pad. Because the geometry of this pad is known, it is possible to infer the MAV's relative pose from the observed perspective projection.

Instead of avoiding the scaling factor problem by only flying in known environments, one can alternatively try to estimate this factor by means of additional sensors. An example is the remote controlled low-cost quadrotor used in [4], which streams camera images from an on-board monocular camera to a ground computer, running a Simultaneous Localization And Mapping (SLAM) software. This SLAM software is based on Parallel Tracking And Mapping (PTAM) [1], which is a popular open source SLAM implementation known for its robustness and efficiency. The scale is estimated using additional measurements from an on-board ultrasound altimeter. The MAV presented in [5] also employs PTAM for estimating its pose, but here the authors were able to perform all processing on-board. The required scaling factor is obtained by using measurements from an accelerometer and pressure sensor. A similar system, which is also based on PTAM, has been published in [6]. Here the authors estimate the scale using measurements form an IMU.

As mentioned before, the scaling factor problem disappears if stereo vision is used instead of monocular vision. Stereo matching, however, generally has high computational demands. Most less-recent work has thus focused on off-board stereo processing. For example, ground mounted stereo cameras were used in [7], [8] that are focused on a quadrotor MAV, which is remotely controlled by a ground computer. In [9] a forward-facing stereo camera is mounted on a quadrotor MAV, which wirelessly transmits all camera images at a relatively low frame rate. Again, a computer receives those images and then remotely controls the MAV.

Only very recently, the first MAVs have emerged that are able to perform stereo processing on-board. The MAV presented in [10] features a forward-facing stereo camera and runs a dense block matching algorithm with a resolution of 320×240. This MAV was later extended in [11] to use an image resolution of 640×480. While in both cases, the stereo matching results are only used for obstacle avoidance and visual markers are still required for navigation, this limitation was resolved with a further extension of this platform in [12]. However, according to the numbers given for this final revision, stereo processing only runs at a relatively low frame rate of just 5 Hz.

One example for a downward-facing stereo camera and on-board stereo processing can be found in [13]. For this MAV, the authors use a dense correlation based stereo algorithm which runs at a very low frame rate of just 3 Hz. The movement of the MAV is calculated using visual odometry and the resulting data is fused with further odometry data gained from an on-board laser scanner. Other work that is worth mentioning includes the lighter-than-air MAV presented in [14], which is equipped with three fisheye cameras, of which two are arranged in a stereo configuration. However, no stereo matching is being performed, but rather the imagery of each camera is tracked individually using PTAM. After tracking, the data from all cameras is fused using a pose alignment step. For this MAV, all processing is performed off-board and no autonomous control has been demonstrated.

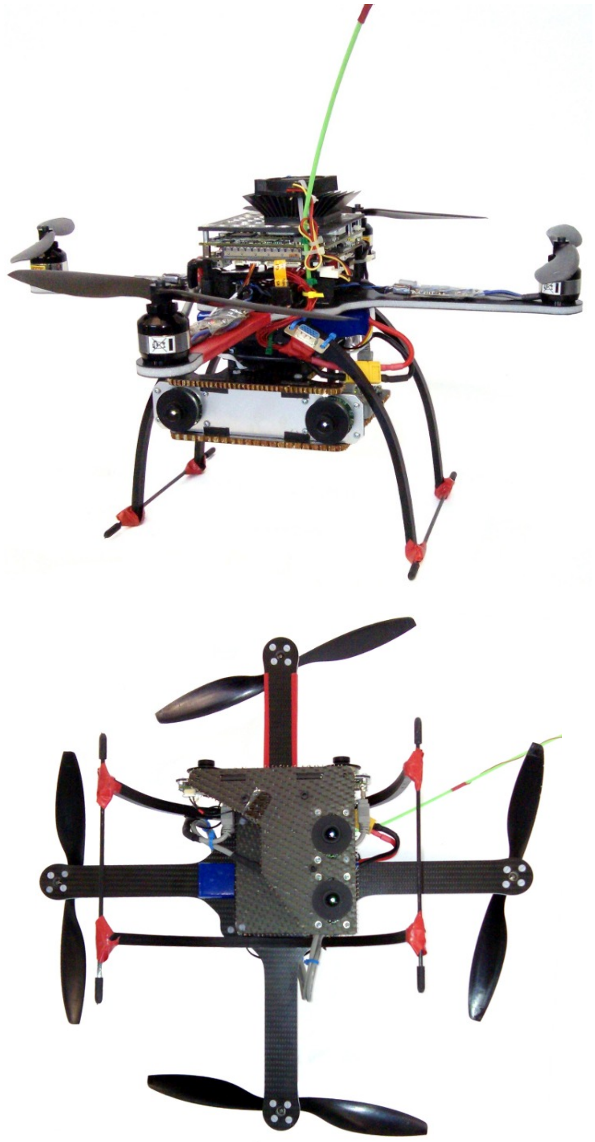


Fig. 1: Our quadrotor MAV seen from front and bottom.

What the presented MAVs that perform stereo matching have in common is that they all employ dense stereo algorithms. There exists, however, at least one example for sparse stereo matching with a forward-facing camera pair [15]. Here, the sparsely matched features are used for a modified version of the SLAM method presented in [16], which itself is an extension of PTAM that incorporates depth information. Because sparse stereo matching is much faster than any dense algorithm, this MAV can maintain a high pose estimation rate of 30 Hz. This high performance can be credited to the efficiency of PTAM as well as to the efficiency of the used sparse stereo algorithm, which was previously published in [17]. Because this method appears to be much faster than all other existing approaches, it makes the most promising base for creating an MAV capable of simultaneously processing the imagery of two stereo cameras.

III. Hardware Overview

A front- and bottom-view of our quadrotor MAV can be seen in Fig. 1. This quadrotor is based on the open source and open hardware PIXHAWK platform, which was developed by the ETH Zürich [11]. We equipped this quadrotor with four USB cameras in two stereo configurations. Two cameras are facing forward with a baseline of 11 cm, while the remaining two cameras are facing downwards with a baseline of 5 cm. We operate the forward-facing cameras with a frame rate of 30 Hz, and the downward-facing cameras with 15 Hz. This unequal frame rate has been chosen in order to reduce the computational requirements. Even though the cameras are operated with different frame rates, they are still synchronized with the downward-facing cameras skipping every other frame. All cameras have a gray-scale image sensor with a resolution of 640 × 480.

The MAV features an on-board computer with an Intel Core 2 Duo CPU, running at 1.86 GHz. All cameras are connected to this computer, which executes all image processing and motion estimation software. In addition to this on-board computer, the MAV is also equipped with a microcontroller, dedicated to all low-level control tasks. The microcontroller board, which also includes an IMU, is connected through an I^{2}C bus with the main on-board computer. Through this bus, the on-board computer transmits high-level control instructions to the microcontroller, which in return transmits the current IMU measurements.

IV. Processing of Forward-Facing Cameras

For the forward-facing cameras we use a method that largely matches the processing pipeline described in [15]. We improved the performance of this method mainly through code level optimizations and by resolving one problem in the original PTAM code, which in case of small maps causes Bundle Adjustment to be executed too frequently. We have made several extensions in order to allow the conjoint use with our processing methods for the downward-facing cameras. Those extensions will be explained later, when their necessity becomes apparent. For completeness, we provide a summary of the original method from [15] at this point.

As this method is based on sparse stereo matching, the first processing step is to apply a feature detector and a sparse stereo matching algorithm. The two algorithms published in [17], for which an efficient open source implementation is available, are used for both of these tasks. Feature detection is, however, extended to include a scale space and an upper bound for the total number of features. We set our bound to 800 features, which is less than the 1000 features used in [15]. An example for the performance of this stereo method during indoor flight of our MAV is given in Fig. 2

The successfully matched features are used for estimating the current MAV pose. A SLAM system is employed for this task, which is based on the method proposed in [16]. This method is an adaptation of PTAM that incorporates depth information, which we receive from stereo matching. This SLAM system has, however, been modified in several ways. Most importantly, the map is continuously cleared of old keyframes, such that only a small set of most recent keyframes persist. This enables us to achieve a constant processing performance, which is crucial for the control of an MAV. Only keeping the most recent keyframes, however, does no longer resemble real SLAM, as this leads to the absence of a global map. The resulting approach can be regarded as a compromise between SLAM and plain visual odometry. To avoid confusion, we will be referring to this method as *local SLAM* for the rest of this paper.

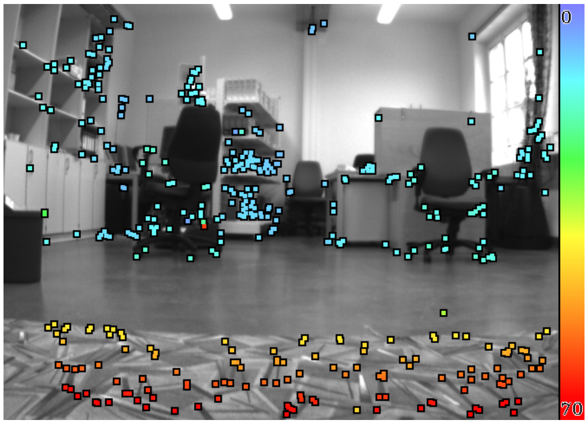


Fig. 2: Example for on-board stereo matching performance.

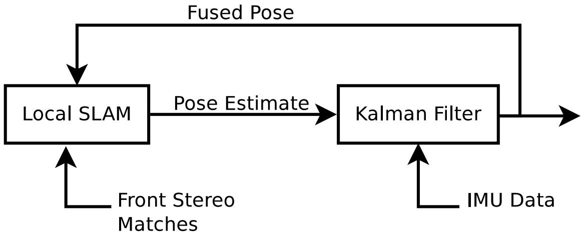


Fig. 3: Schematics of sensor fusion for only two cameras.

The received pose is fused with measurements from the IMU. The basic schematics of the sensor fusion are depicted in Fig. 3. For this task, the open source ROS package `imu_filter`[1] is employed, which is based on an extended Kalman filter. Here, the IMU measurements are used for performing the Kalman prediction, while the Kalman correction is then executed with the estimated pose. Unlike in [15], we process the IMU measurements at a higher rate of 100 Hz.

For tracking a new frame, the local SLAM system requires an estimate for the current pose, which is obtained through a motion model. This motion model extrapolates the fused pose that is fed back to local SLAM after sensor fusion. Rotation, however, is predicted by using an Efficient Second Order Minimization (ESM) [18] based image alignment method, which can be found in the original PTAM implementation.

[1] See `http://ros.org/wiki/imu_filter`

V. Problems with Forward-Facing Cameras

The method we have discussed above can be used to achieve autonomous hovering of an MAV, as has been demonstrated in [15]. However, there are some problems with using this approach as is. In this section we provide a qualitative overview of these issues. Quantitative examples for some of these problems can be found in Sec. X.

One key issue is the potential drift of the estimated position and orientation. If the orientation estimated by local SLAM is used for controlling the quadrotor, any errors in the estimated pitch and roll angles will have a disrupting impact on the flight stability. In [15] the MAV was controlled with the original PIXHAWK flight controller, which only relies on IMU measurements for determining the current attitude. It would be very preferable to use the more accurate vision estimated orientation instead. However, this would make the handling of orientation errors even more important.

Not only orientation drift can be problematic, but also drift errors for the estimated position are an issue. If the MAV is programmed to fly on a preset track, a position deviation would cause the MAV to leave this track, which can potentially lead to a crash. But even if the MAV performs on-board path planning in consideration of the perceived environment, position drift can still cause troubles. For example, the MAV presented in [12] performs such autonomous on-board path planning, but this happens only in two dimensions with a fixed flying altitude. If such an MAV does not have any other means for perceiving the current altitude, it is unable to react on position drifts in the vertical direction.

The local SLAM method also has difficulties with yaw rotations. This was more severe for the original PTAM version, which just processes imagery of one monocular camera. Because no triangulation can be performed for rotation-only movements, PTAM is not able to obtain reliable depth measurements in this case. The local SLAM method we use obtains its depth information from stereo vision, which should make yaw rotations less problematic. However, fast yaw rotations still lead to large image movements, which can result in bad tracking accuracy or even tracking failure.

Finally, if tracking ever fails, recovery can only occur if the camera still depicts a scene that has been well observed by at least one existing keyframe. Since the MAV is likely to be on a flying trajectory, we cannot expect that this is the case. Even if the camera hasn't moved much since the previous keyframe, recovery might still fail which can lead to a random new position. Thus, recovery needs to be improved if we want to achieve robust flight.

The problems we have described so far can be solved or at least be reduced, if we employ a pair of downward-facing cameras in addition to the already used forward-facing ones. How exactly this can be achieved will be discussed in the following sections.

VI. Processing of Downward-Facing Cameras

Our processing method for the downward-facing cameras differs fundamentally from the method presented above. At the beginning, however, there is again stereo matching, for which we employ the same technique previously used for the forward-facing cameras. This time, however, we set the maximum feature count to 300 and process images at a rate of just 15 Hz. This is done in order to save computing resources, as the method is less sensitive to feature count and image rate, than the used local SLAM system.

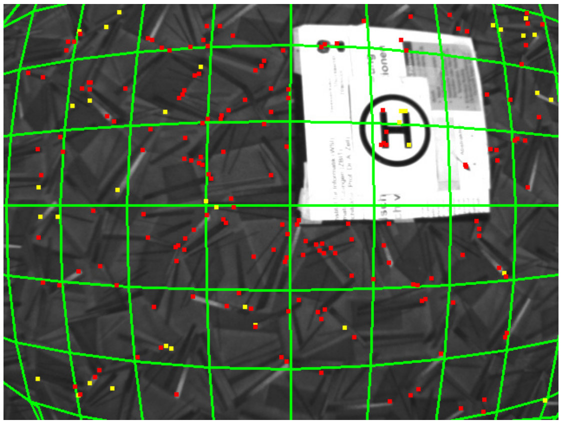


Fig. 4: Example for on-board ground plane detection.

A. Ground Plane Detection

If we assume that the ground is flat, then all 3D points received from stereo matching are expected to lie in the same geometric plane. We obtain an estimate for this ground plane with a RANdom SAmpling Consensus (RANSAC) based plane estimator. Given a plane equation in the form of

$$ax + by + cz + d = 0, \tag{1}$$

we can extract the height h, pitch angle Θ and roll angle Φ with the following equations:

$$h = \frac{-d}{b} \tag{2}$$

$$\Theta = \tan^{-1}\left(\frac{-a}{b}\right) \tag{3}$$

$$\Phi = \tan^{-1}\left(\frac{-c}{b}\right) \tag{4}$$

Unlike the pose estimates of the local SLAM system, these measurements are absolute. Hence, they are not prone to drift or erroneous offsets, which is why we expect those measurements to increase the overall accuracy in a combined system. We estimate the variances of those measurements by using a sampling based approach. For the height variance σ_h this happens by calculating the distance between the plane model and each point that was selected as inlier by the RANSAC method. The variance of the mean distance is then our estimate for σ_h. For the angular variances σ_Θ and σ_Φ, we first shuffle the set of inliers such that three consecutive points always have a large distance to each other. We then repeatedly draw three such points and use them to calculate samples for Θ and Φ. The variances of the means of those samples are then our final estimate.

Finally, we apply a simple outlier rejection based on the previously detected plane model. An example on how our

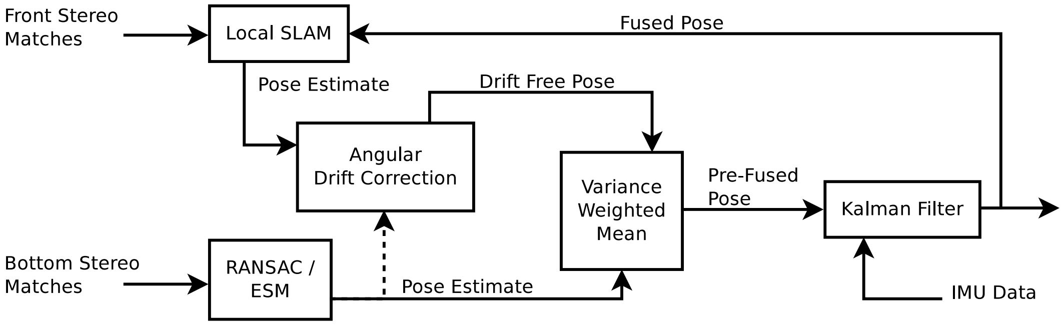


Fig. 5: Schematics of sensor fusion with forward-facing and downward-facing cameras.

method performs for ground plane detection can be found in Fig. 4. Here, the plane has been projected back into the unrectified camera image. The red points indicate features that were selected as inliers by the RANSAC plane estimator, while yellow points were classified as outliers.

B. Frame-to-Frame Tracking

It is unfortunately not possible to infer horizontal displacement or yaw rotation from the detected plane. For measuring these quantities, we hence need to apply a different method. In our case, we have selected an approach based on frame-to-frame tracking. Because we assume a flat ground, horizontal displacements and yaw rotations will result in an affine image transformation. This transformation consists of a 2d-translation and an in-plane rotation. We hence attempt to find the transformation that aligns a previously captured camera frame to the current camera image.

For this task we chose the previously mentioned ESM-algorithm, which uses a homography to align two images. This happens by iteratively applying transformations to an initial homography until the algorithm converges to a steady solution. In our case, we limit ourselves to a homography consisting of translations and in-plane rotations. Even though ESM is an efficient method, finding the transformation between two full-resolution frames is still very time consuming. Hence, we perform this step with two very low-resolution sub-sampled frames. In fact, we use a resolution of just 80×60 pixels. This number might seem small, but because ESM works well at the sub-pixel level, we still receive sufficiently accurate results.

To have a large field of view, we use lenses with small focal lengths on the downward-facing cameras. This, however, leads to strong lens distortions that disrupt the frame-to-frame tracking. Hence, we first perform image rectification, which can be combined with the required sub-sampling into one single image transformation. This transformation is much faster than an individual rectification at full image resolution, and also avoids unnecessary blur.

Once the ESM algorithm has converged, we extract our measurements from the found homography. The translation vector matches the fourth column of the homography matrix. The rotation, however, cannot as easily be extracted without applying a homography decomposition. Thus, we apply the simple approach of transforming a distant point with the found homography, and then measuring the angle towards the point's initial location. Using the height that we received from the detected ground plane and the known camera parameters, we can convert the translation vector from pixel to world coordinates. Together with the yaw rotation and the measurements received from the detected ground plane, we obtain a full 6DoF estimate of the current MAV pose.

VII. Sensor Fusion

With the methods described above we receive two independent estimates for the MAV pose: one from the forward-facing, and one from the downward-facing camera pair. These two estimates need to be unified into one single pose estimate, which happens during sensor fusion. For this purpose we can complement the sensor fusion we have discussed earlier. The schematics of our extended sensor fusion are shown in Fig. 5. For now we ignore the block labeled "Angular Drift Correction", as this part will be discussed in detail later on.

Thanks to camera synchronization, measurements will be from exactly the same point of time, if received from both camera pairs. In this case, we fuse both poses before applying the Kalman filter. This happens by calculating the variance weighted mean, which delivers us a maximum likelihood estimate for the current pose. The reason why we fuse both poses before applying the Kalman filter is that we want to avoid giving preference to either of them. If both pose estimates would be processed individually, the pose processed last would have the higher influence on the Kalman filter. This is usually the pose obtained through the forward-facing cameras, as it requires more time to be computed.

If only one pose is available due to the bottom cameras skipping one frame, or because one method fails to deliver a reliable pose estimate, the variance weighted mean is avoided and the single pose estimate is processed by the Kalman filter as is. In any case, the fused pose is passed on to the low-level flight controller. Unlike the original PIXHAWK flight controller, we use the fused pose for attitude estimation, rather than relying on IMU measurements.

VIII. Drift Correction

Preliminary experiments with the method presented so far have shown that there are still problems with flight stability. This can mostly be credited to accumulated errors that lead to unwanted drift. Two such error sources have been identified and we have been able to compensate them by using additional processing steps.

A. Map Drift

One major source of error is the map generated by the local SLAM system. Once a keyframe has been created, its position is only altered by Bundle Adjustment, which generally only performs minor corrective changes. This means that if a keyframe has been created at an incorrect position, this error will not be corrected until the keyframe is discarded. Hence, all pose estimates that are obtained by matching against this keyframe will be inaccurate.

The downward-facing cameras deliver us absolute measurements for height, roll and pitch. With those absolute measurements we should be able to at least partially correct inaccurate keyframes. The fused position, which contains contributions from those absolute measurements, is already fed back to the local SLAM method (see Fig. 5). However, so far the fused pose is only used for motion prediction, which does not have an influence on the existing map.

It is thus necessary to correct the pose of exiting keyframes. We do this by applying a global transformation to the entire map. This transformation is chosen such that it compensates the difference between the last pose estimated by the local SLAM system and the final pose estimate after sensor fusion. If $\boldsymbol{T_s}$ is the transformation matrix for the pose estimated by local SLAM, and $\boldsymbol{T_f}$ is the transformation matrix after sensor fusion, then the matrix product $\boldsymbol{T_f}^{-1} \cdot \boldsymbol{T_s}$ represents the transformation that we required to map $\boldsymbol{T_s}$ to $\boldsymbol{T_f}$. We hence define our corrective transformation $\boldsymbol{T_c}$ as:

$$\boldsymbol{T_c} = \lambda(\boldsymbol{T_f}^{-1} \cdot \boldsymbol{T_s}) \tag{5}$$

Here, the transformation is scaled with the weighting factor λ. This weight is set to a small value (we use a value of 0.05), such that only small corrective steps are performed. Any drift or error will thus be gradually reduced over several frames. Further, we force the horizontal displacement of the corrective transformation to 0. Because there is no sensor that delivers absolute measurements of the horizontal position, we prefer to keep the position estimated by local SLAM instead.

B. Angular Drift

Although the previously described drift correction works well for correcting the height of a keyframe, its performance is generally poor for roll and pitch errors. This can be explained by examining the variances used during sensor fusion. While the height measurements received from the downward-facing cameras are more accurate than the measurements received from local SLAM, the variances for the measured roll and pitch angles are several orders of magnitude larger. This means that roll and pitch measurements from the downward-facing cameras are mostly ignored during sensor fusion.

Unlike local SLAM, however, the downward facing cameras provide an absolute measurement, which is why we do not want do disregard this information. We solve this problem by introducing an additional processing step during sensor fusion, which has been labeled "Angular Drift Correction" in Fig. 5. In this step, we try to estimate the angular drift of the local SLAM pose and correct it before sensor fusion starts. Because the angular measurements from the downward-facing cameras are considerably noisy, we employ an additional Kalman filter for this task. This Kalman filter tracks the difference between the orientation estimate gained from local SLAM and from the downward-facing cameras. We represent the orientation as quaternions, which matches the representation used in the entire sensor fusion pipeline.

If we are able to correct the angular drift, then the pose received after sensor fusion should contain the correct orientation. We know that the fused pose is fed back to local SLAM, where it is used to correct the map drift with respect to the weight λ. Hence, we can also expect that the angular drift will be reduced in the next frame. This knowledge can be incorporated into the model of our Kalman filter. We assume that the arithmetic difference between the two orientation quaternions Δ_q reduces to $\Delta_q \cdot (1 - \lambda)$ from one frame to another. If we ignore all other influences on the orientation drift, then we arrive at the following state transition matrix:

$$\boldsymbol{F_k} = \begin{pmatrix} 1-\lambda & 0 & 0 & 0 \\ 0 & 1-\lambda & 0 & 0 \\ 0 & 0 & 1-\lambda & 0 \\ 0 & 0 & 0 & 1-\lambda \end{pmatrix} \tag{6}$$

The filtered quaternion difference is then added to the orientation quaternion from local SLAM, which effectively removes orientation drift. However, we further adapt this pose by restoring the yaw rotation to its uncorrected value, as there are no absolute measurements for the yaw angle.

IX. Recovery

The last remaining problem that needs to be solved is recovery of the local SLAM system in case of tracking failure. As discussed previously, the recovery approach employed by PTAM does not work well for our application. We hence use a different technique that makes use of the redundant information available from both camera pairs. Even when the local SLAM method fails, we still receive a full 6DoF pose estimate from the downward-facing cameras. Thus, the pose of the MAV is still known but with a degraded accuracy. Nevertheless, we should be able to maintain control of the MAV until local SLAM has recovered.

If tracking fails, we force the local SLAM pose to the current output of the sensor fusion. In this case, the fused pose is only obtained through measurements of the downward-facing cameras and the IMU. This pose will, however, not match the current map of the local SLAM system, which prevents the system from recovering by itself. We hence discard the entire map and begin mapping from scratch. We start by adding the current frame at the currently available fused pose. The system should thus quickly recover once the cause of the error has disappeared. Usually, tracking failures result from quick camera movements. Hence, once

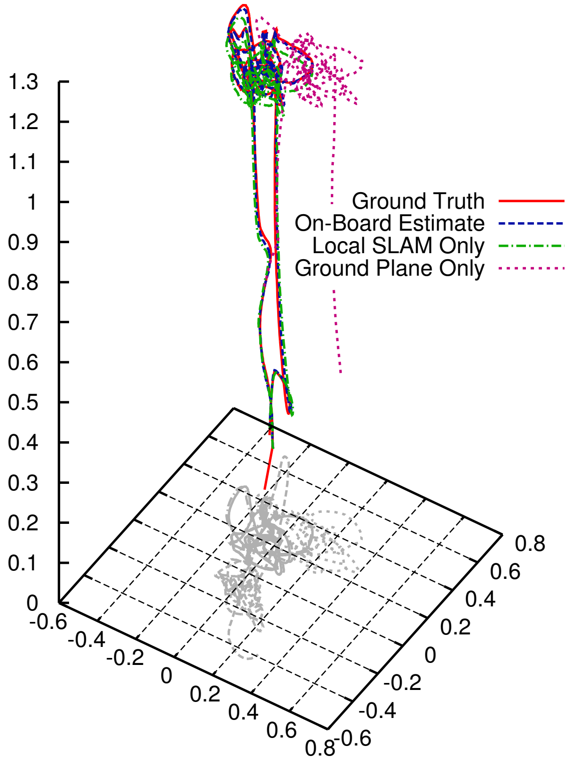


Fig. 6: Ground truth and estimated position during autonomous take-off, hovering and landing. Scale is in meters.

the MAV has stabilized itself by using the less error prone pose estimates from the downward-facing cameras, local SLAM will continue functioning.

X. Evaluation

We have conducted several experiments to evaluate the quality of the proposed MAV design. All flying experiments took place in the same indoor lab environment. We covered the floor with a texture rich carpet in order to provide sufficient features for the downward-facing cameras. Examples for the scene observed by the forward- and downward-facing cameras are shown in Fig. 2 and 4. For all experiments we recorded ground truth motion information with an OptiTrac optical tracking system.

A. Hovering

In our first experiment we instructed the quadrotor to hover at a height of approximately 1 m for one minute, with take-off and landing being performed autonomously. This should allow an assessment of the MAV's flight stability and demonstrate its general operativeness. During flight, the MAV recorded all sensor data (i.e. camera imagery and IMU measurements), allowing us to re-process all test-runs offline.

The ground truth position and the position estimate obtained by the on-board software are depicted in Fig. 6. This figure contains yet two more curves, which are the position estimates received when offline processing the recorded data with only the local SLAM system or only our ground-plane-based pose estimation. By plotting these offline results, we can compare how the MAV would have behaved, if it had been equipped with only two cameras. All plotted tracks have been aligned such that their starting position and orientation match closely. For this purpose we iteratively performed an error minimization for each position coordinate and the yaw rotation for the first 0.5 s of each track.

Method	RMSE	Avg. Error
On-Board Estimate	1.41 cm	1.44 cm
Local SLAM only	3.14 cm	3.15 cm
Ground Plane Only	26.2 cm	23.7 cm

TABLE I: Position estimation errors for the examined processing methods.

The slow take-off and landing in this experiment, as well as the stable hovering position, are also an easy challenge for the local-SLAM-only test run. Hence, the corresponding curve and the curve for our on-board estimate both closely match the recorded ground truth. The ground-plane-only based pose estimate, however, shows accurate height but exhibits high horizontal drift. While in this case, the absolute height can be measured, the horizontal position is only obtained through frame-to-frame tracking, which is particularly prone to error accumulation.

We can quantify the deviation from the ground truth by examining the Euclidean distance between the estimated and ground truth position. If we do this for all position estimates, we receive the errors listed in Table I. The table reveals that the errors for our on-board estimates are less than half as large as the errors we receive when re-processing the sequence with the local-SLAM-only system. Much of this improvement can be credited to the more accurate height that we obtain with the combined approach. As we have already anticipated from Fig. 6, the errors received with the ground-plane-only based method are much higher than for the other two test runs.

The more interesting measure, however, is the MAV's ability to keep its hovering location. To remain comparable, we apply the same measure as used in [15]. This means that we calculate the position error with respect to the average position during hovering. In our case we receive an average error of 11.0 cm and a RMSE of 12.6 cm. Compared to the values reported in [15] (average error of 26 cm and RMSE of 32 cm), our position errors are much smaller, meaning that our MAV can keep its hovering location more precisely. We believe that much of this performance improvement can be credited to our modified flight controller, which now obtains its attitude estimate from our on-board vision software.

B. Drift Compensation

The next interesting characteristic is the performance of our drift correction methods. Because it is difficult to evaluate the drift correction in a flying experiment, we decided to simulate drift errors instead. We take the sensor data recorded during an autonomous hovering flight, and re-process this data offline. While the MAV is hovering, we force an erroneous orientation and height into the system. We do this by disturbing the output of the sensor fusion for a short period of time. During this time, we keep on

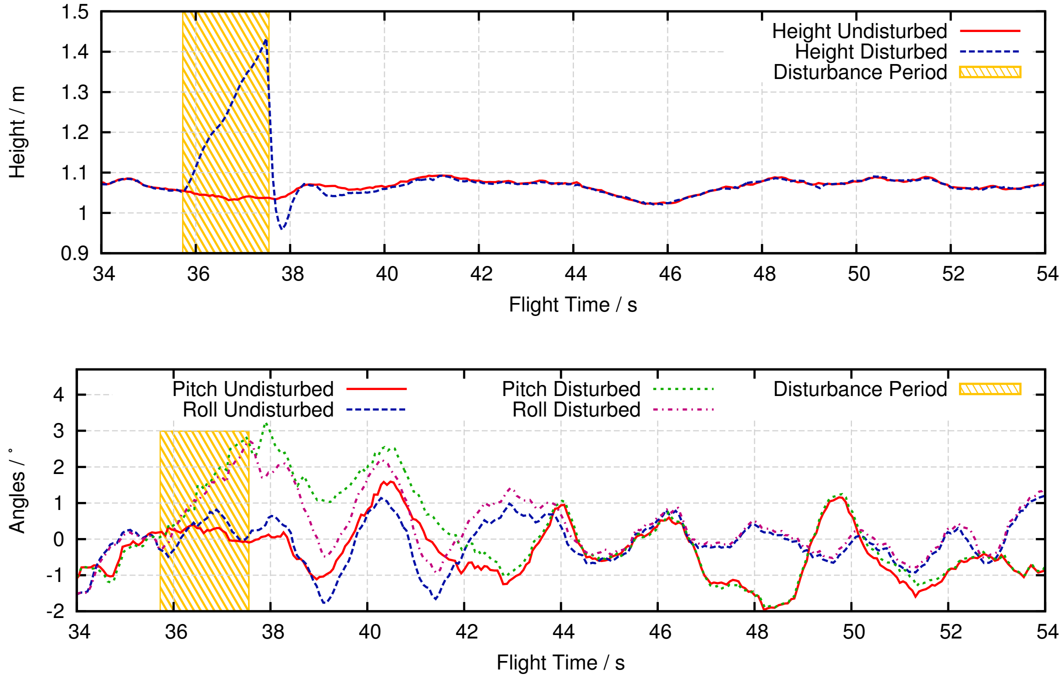


Fig. 7: Recovery of (top) height estimates and (bottom) roll- and pitch-angle after forceful disturbance.

applying an erroneous rotation and vertical translation to the fused pose, which is being fed back to local SLAM. This disturbance forces local SLAM into recovery, which means that mapping starts again from scratch.

The height recorded in this experiment is plotted in the top half of Fig. 7. The disturbance is applied during the highlighted section. For comparison, we also included the undisturbed height that was estimated on-board during autonomous hovering. We see that the height from both recordings diverge once the disturbance is applied. Once the disturbance period ends, however, the height measurements quickly converge again to the undisturbed on-board estimates. Similarly, the disturbed and undisturbed roll and pitch angles are shown in the bottom half of Fig. 7. Again, the angular measurements converge to the undisturbed estimates after the disturbance period has finished. This successfully demonstrates the functioning of our drift correction methods.

C. *Yaw Rotations*

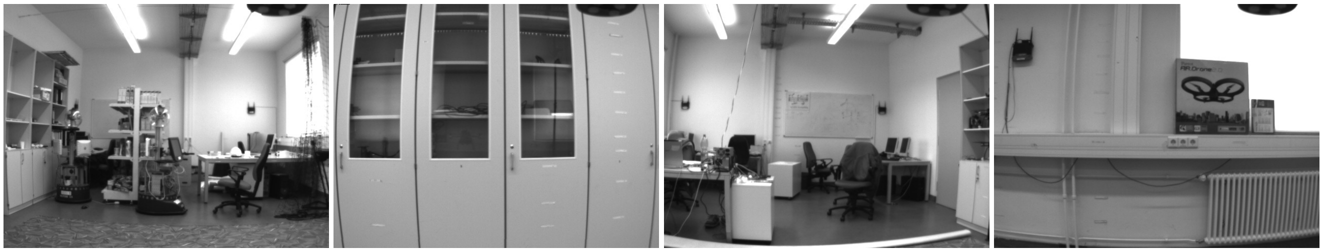
While our previous hovering experiment is also a feasible task with only two cameras and local SLAM, the situation is different if we encounter yaw rotations. As we have discussed earlier, observing fast yaw rotations with the forward-facing cameras is particularly challenging. Thus, we expect that our MAV will benefit much from our additional downward-facing cameras. For putting this assumption to a test, we let our MAV perform a 360° yaw rotation. This rotation was divided into four separate 90° turns, for which our MAV required an average time of 2.3 s each. After each turn, the MAV waited for itself to stabilize and then hovered for 5 seconds before continuing with the next turn. An example for the scene observed by the forward-facing cameras after each turn is shown in Fig. 8.

Figure 10 contains the recorded ground truth and on-board position estimates for a typical test run of this experiment. We again re-processed the recorded camera imagery and IMU measurements offline with a local-SLAM-only and ground-plane-only version of our software system, and included the results in Fig. 10. In this figure we see that despite the yaw rotations, the MAV is able to maintain an accurate estimate of its current position. The ground-plane-only test run again shows the already observed behavior of accurate height estimates but strong horizontal drift.

The position estimated by the local-SLAM-only version, on the other hand, shoots off in a random direction after the first 90° turn. Please note that the diagram has been truncated and that the position estimation continues to show the same erroneous behavior for each of the four turns. In fact, we have never been able to obtain a valid position estimate beyond the first turn for any test-run with the local-SLAM-only version. If the MAV had used this erroneous position estimate for navigation, this would have inevitably led to a crash.

The recorded and estimated yaw rotations are depicted in Fig. 9. In this diagram we see that the yaw rotation estimated with our on-board method closely follows the ground truth, while the ground-plane-only version follows the ground truth less accurately. The local-SLAM-only version, starts deviating significantly after the first turn, which matches the observation from Fig. 10.

The good performance of our method can in large parts be credited to our recovery strategy. In fact, recovery of the local SLAM method was performed once during each turn. Because the more rotation-robust pose from the downward-facing cameras is used during recovery, our MAV is able to keep an accurate pose estimate throughout the experiment.



(a) 0°

(b) 90°

(c) 180°

(d) 270°

Fig. 8: Scene observed by the forward-facing cameras during 360° yaw rotation.

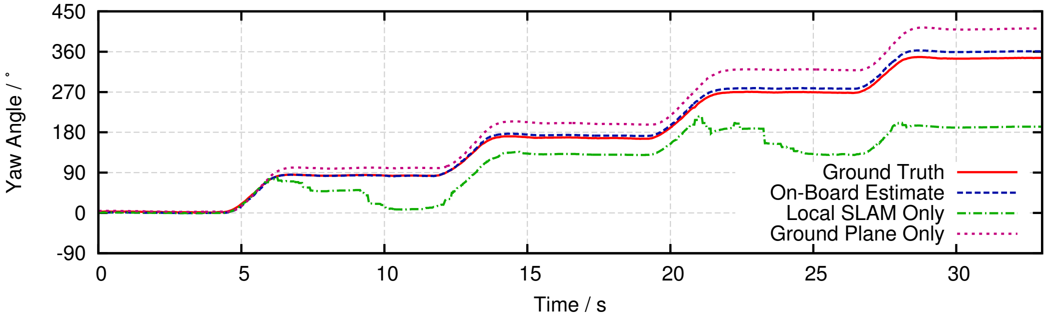


Fig. 9: Yaw angles measured during 360° yaw rotation.

D. Square Flight

In a final experiment we let the MAV fly a horizontal square-shape, with an edge length of 1 m. Please note that this is just 2.5 times the rotor-to-rotor distance. Therefore, this task requires precise rotor control and poses a considerable challenge for the MAV. The MAV first took-off autonomously and then approached each corner of the square twice before landing again. It hovered at each corner for 5 seconds before continuing its trajectory. A top view of the recorded ground truth flight path is given in Fig. 11. For clarity, we have omitted the MAV path during take-off and landing. Further, we also included an idealized reference track, which we aligned against the starting pose of the MAV. These results shall only serve as a qualitative impression on the performance of our MAV, which is why we omit a quantitative analysis.

XI. Conclusions

In this paper we have proposed a novel design for an autonomous quadrotor MAV. In a GPS-denied space, the key problem of autonomous flight is self-localization. We have solved this problem for our MAV by employing four cameras in two stereo configurations. We are able to perform stereo processing for both camera pairs on-board the MAV in real time. To our knowledge, on-board stereo processing with more than two cameras has not been demonstrated before.

We have used a dedicated processing technique for the stereo matching results of each camera pair. While the imagery of the forward-facing cameras is processed with an approach based on stereo SLAM, the downward-facing cameras can be processed more efficiently if we assume that the ground is flat. For this camera pair we thus use a method based on RANSAC plane detection and ESM homography estimation. Combining the pose estimates we obtain from each camera pair was one of the main challenges. While a sensor fusion can be applied to receive the current pose estimate, we also need to update the map created by the local SLAM method. For this task, we included the two presented drift correction techniques.

The resulting MAV has been evaluated in several flight experiments. In a hovering test, the MAV took-off autonomously, hovered at a given height and then landed again by itself. It was able to keep its hovering location with an average error of only 11.0 cm. By re-processing the collected sensor data with only the forward-facing cameras, we have been able to show that the downward-facing cameras indeed increase the pose estimation accuracy.

In further experiments we have demonstrated the robustness of our MAV design. Pose estimation is able to automatically recover when an erroneous height and orientation are forcefully introduced into the system. Further, we have shown that our MAV is able to successfully fly a 360° yaw rotation. This is a difficult maneuver, for which we received very bad results when using only the forward-facing cameras. Our MAV was able to perform this operation thanks to the availability of a redundant pose estimate from the downward-facing cameras. Further, our tracking recovery method played an important role in this experiment, which is able to perform a re-initialization of the local SLAM system.

With this work we have shown that by simultaneously using both, a forward-facing and a downward-facing camera pair, we can significantly increase accuracy and robustness of the MAV navigation. Forward- and downward-facing cameras have different strengths and weaknesses, and are

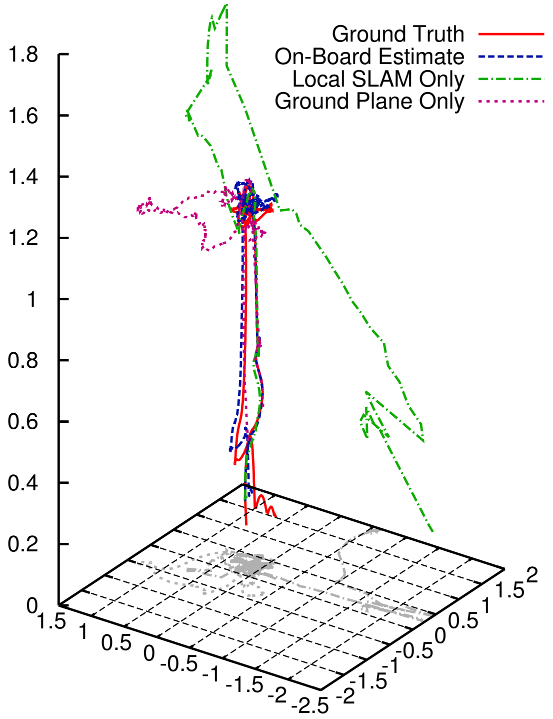


Fig. 10: Ground truth and estimated position during 360° yaw rotation. Scale is in meters.

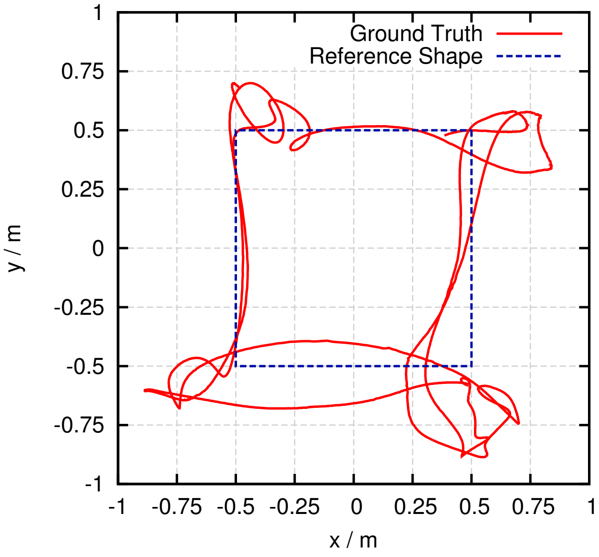


Fig. 11: Flight of a horizontal rectangle shape.

thus able to complement each other. Using more cameras also adds redundancy to the system, which leads to an increased fail-safety. There exist, however, situations in which the performance of all cameras can be impeded, such as low lighting, fog or insufficient texture. To further increase redundancy, we would thus propose to equip the MAV with additional sensors, such as laser scanners, ultrasonic transceivers or a depth camera. The very limited payload of most MAVs, however, does not allow us to carry too much sensory equipment. Hence, such extensions will have to wait until lighter and smaller sensors are available.

References

[1] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 1–10.

[2] G. P. Tournier, M. Valenti, J. How, and E. Feron, "Estimation and Control of a Quadrotor Vehicle Using Monocular Vision and Moiré Patterns," in *In AIAA Guidance, Navigation and Control Conference*, 2006, pp. 2006–6711.

[3] S. Yang, S. A. Scherer, and A. Zell, "An Onboard Monocular Vision System for Autonomous Takeoff, Hovering and Landing of a Micro Aerial Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 69, pp. 499–515, 2012.

[4] J. Engel, J. Sturm, and D. Cremers, "Camera-Based Navigation of a Low-Cost Quadrocopter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 2815–2821.

[5] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown in- and Outdoor Environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3056–3063.

[6] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM–Based Navigation for Autonomous Micro Helicopters in GPS-Denied Environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[7] M. Achtelik, T. Zhang, K. Kuhnlenz, and M. Buss, "Visual Tracking and Control of a Quadcopter Using a Stereo Camera System and Inertial Sensors," in *International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2009, pp. 2863–2869.

[8] D. Pebrianti, F. Kendoul, S. Azrad, W. Wang, and K. Nonami, "Autonomous Hovering and Landing of a Quad-Rotor Micro Aerial Vehicle by Means of on Ground Stereo Vision System," *Journal of System Design and Dynamics*, vol. 4, no. 2, pp. 269–284, 2010.

[9] L. R. G. Carrillo, A. E. D. López, R. Lozano, and C. Pégard, "Combining Stereo Vision and Inertial Navigation System for a Quad-Rotor UAV," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 373–387, 2012.

[10] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous Obstacle Avoidance and Maneuvering on a Vision-Guided MAV Using On-Board Processing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2472–2477.

[11] L. Meier, P. Tanskanen, L. Heng, G. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision," *Autonomous Robots*, pp. 1–19, 2012.

[12] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4557–4564.

[13] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[14] A. Harmat, I. Sharf, and M. Trentini, "Parallel Tracking and Mapping with Multiple Cameras on an Unmanned Aerial Vehicle," in *International Conference on Intelligent Robotics and Applications (ICIRA)*, vol. 1. Springer, 2012, pp. 421–432.

[15] K. Schauwecker, N. R. Ke, S. A. Scherer, and A. Zell, "Markerless Visual Control of a Quad-Rotor Micro Aerial Vehicle by Means of On-Board Stereo Processing," in *Autonomous Mobile System Conference (AMS)*. Springer, 2012, pp. 11–20.

[16] S. A. Scherer, D. Dube, and A. Zell, "Using Depth in Visual Simultaneous Localisation and Mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 5216–5221.

[17] K. Schauwecker, R. Klette, and A. Zell, "A New Feature Detector and Stereo Matching Method for Accurate High-Performance Sparse Stereo Matching," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5171–5176.

[18] S. Benhimane and E. Malis, "Real-Time Image-Based Tracking of Planes Using Efficient Second-Order Minimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2004, pp. 943–948.