Towards autonomous navigation of miniature UAV

Roland Brockers
Jet Propulsion Laboratory
brockers@jpl.nasa.gov

Martin Humenberger
Austrian Institute of Technology
martin.humenberger@ait.ac.at

Stephan Weiss
Jet Propulsion Laboratory
Stephan.Weiss@ieee.org

Larry Matthies
Jet Propulsion Laboratory
lhm@jpl.nasa.gov

Abstract

Micro air vehicles such as miniature rotorcrafts require high-precision and fast localization updates for their control, but cannot carry large payloads. Therefore, only small and light-weight sensors and processing units can be deployed on such platforms, favoring vision-based solutions that use light weight cameras and run on small embedded computing platforms. In this paper, we propose a navigation framework to provide a small quadrotor UAV with accurate state estimation for high speed control including 6DoF pose and sensor self-calibration. Our method allows very fast deployment without prior calibration procedures literally rendering the vehicle a throw-and-go *platform. Additionally, we demonstrate hazard-avoiding autonomous landing to showcase a high-level navigation capability that relies on the low-level pose estimation results and is executed on the same embedded platform. We explain our hardware-specific implementation on a 12g processing unit and show real-world end-to-end results.*

1. Introduction

Miniature rotorcraft platforms have several advantages in exploration and reconnaissance missions since they can operate in highly cluttered environments (forest, close to the ground) or confined spaces (indoors, collapsed buildings, caves) and offer stealth from their small size. But in order to operate these platforms safely, fast and accurate pose estimation that is independent from external sensors (e.g. GPS) is needed for control.

Literature has shown that a viable solution for GPS independent pose estimation is to use visual and inertial sensors [6, 11]. However, a major algorithmic challenge is to process sensor information at high rate to provide vehicle control and higher level tasks with real-time position information and vehicle states. Since micro rotorcrafts can only carry a few grams of payload including batteries, this has to be accomplished with a very small weight and power budget. Consequently, only light-weight sensors and processing units can be used on such platforms, favoring vision-based pose estimation solutions that use small light-weight cameras and MEMS (microelectromechanical systems) inertial sensors. As recent developments in multi-core smartphone processors are driven by the same size, weight, and power (SWaP) constraints, micro aerial vehicles (MAVs) can directly benefit from new products in this area that provide more computational resources at lower power budgets and low weight, enabling miniaturization of aerial platforms that are able to perform navigation tasks fully autonomously. Additionally, novel algorithmic implementations with minimal computational complexity, such as presented in this paper, are required.

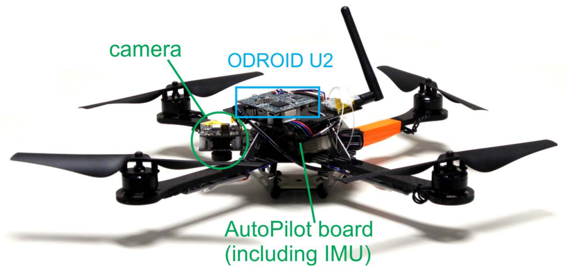

Figure 1. Asctec Hummingbird with Odroid-U2 flight computer mounted on top.

Once pose estimation is available, higher level autonomous navigation tasks which leverage and require this information can be executed. Examples for such tasks are: autonomous landing, ingress, surveillance, exploration, and other. Autonomous landing is especially important not only for safety reasons, but also for mission endurance. Small rotorcrafts inherently suffer from overall short mission endurance, since payload restrictions do not allow carrying large batteries. For surveillance or exploration tasks, endurance can be greatly improved by not requiring the platform to be airborne at all time. Instead, such tasks may even favor a steady quiet observer at a strategic location (e.g. high vantage points like rooftops or on top of telephone poles) - still with the ability to move if required - which also could include re-charging while in sleep mode (e.g. from solar cells). Such a capability allows long term missions, but introduces additional constraints on the algorithm side: First, to eliminate memory and processing power issues, all algorithms need to run at constant complexity with a constant maximal memory footprint, independently of mission length and area, which is particularly true for the pose estimation module. Second, time and impacts with the environment may change sensor extrinsics. To ensure long-term operation, the pose estimator not only has to provide accurate pose information but also needs to continuously self-calibrate the system. Third, since a base station may not be within communication reach, all algorithms have to be self-contained and executed on-board the vehicle.

In this paper, we introduce a novel implementation of a fully self-contained, self-calibrating navigation framework on an embedded, very small form-factor flight computer (12grams) that allows to autonomously control a small rotorcraft UAV (Figure 1) in GPS denied environments. Our framework enables quick deployment (throw-and-go), and performs autonomous landing on elevated surfaces as a high-level navigation task example, which we chose because of three reasons: 1) Landing requires accurate and continuous pose estimation over large scale changes (i.e. from altitude to touch down). Thus, it implicitly shows the capabilities of our underlying estimator. 2) It demonstrates how the information of our robust low-level pose estimator can be further leveraged and integrated to improve subsequent high level tasks. 3) The computational complexity of 3D reconstruction and landing site detection needs to be tractable on such low SWaP devices.

The remainder of the paper is organized as follows: Section 2 gives an overview of our vision-based pose estimation framework including related work and a short introduction of its self-calibration and the underlying filter approach. Section 3 describes our autonomous landing algorithm, and details our approach for 3D reconstruction and landing site detection. Section 4 introduces the on-board implementation, provides a performance evaluation, and presents experimental results, whereas Section 5 concludes the paper.

2. GPS Independent Pose Estimation

Autonomous flight in unknown environments excludes the use of motion capture systems, and using GPS is not always an option since it may be unavailable due to effects such as shadowing or multipath propagation in city-like environments. Therefore, commonly used sensors for pose estimation are stereo [8] and monocular cameras [20] as well as laser scanners [16]. Since heavy sensors cannot be used on low SWaP platforms, monocular, visual-inertial pose estimators might be the most viable choice for MAVs. In our previous work, we demonstrated that pose estimation based on the fusion of visual information from a single camera and inertial measurements from an IMU can be used for simultaneously estimating pose information and system calibration parameters (such as IMU biases and sensor extrinsics). This self-calibrating aspect eliminates the need of pre-launch calibration procedures and renders the system *power-on-and-go*

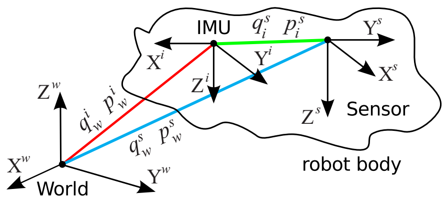

Figure 2. Setup illustrating the robot body with its sensors with respect to a world reference frame. The system state vector is $X = \{p_w^i\ v_w^i\ q_w^i\ b_\omega\ b_a\ \lambda\ p_i^s\ q_i^s\}$ and p_w^i and q_w^i denote the robot's position and attitude in the world frame.

In our current framework, we propose a hierarchical architecture for our vision-front-end with a downward looking camera. Using the camera as a velocity sensor based on inertial-optical flow (IOF), allows to estimate full attitude and drift-free metric distance to the overflown terrain [22], which we use for rapid deployment and emergency maneuvers. For accurate position estimation, we use a keyframe-based visual self localization and mapping (VSLAM) strategy to render the camera into a 6DoF position sensor. The VSLAM approach is based on [7] but tailored to run on our embedded architecture with constant computational complexity by deploying a sliding window, local mapping approach [20]. Whereas our velocity based IOF approach drifts parallel to the terrain our VSLAM approach is locally drift free and better suited for long-term MAV navigation in large outdoor environments – only limited by the battery life-time and not by processing power nor memory. But it requires specific motion for initialization. Since the IOF based approach does not need any particular initialization, it can be used for very quick deployment of the MAV – even by simply throwing it in the air. Thus, for rapid deployment, the thrown MAV will quickly stabilize itself with our IOF approach, and then autonomously initialize our VSLAM approach as shown in [21] – rendering it a *throw-and-go* GPS independent system.

Both, the pose of the VSLAM algorithm as well as the output of the IOF, are fused with the measurements of an IMU using an Extended Kalman Filter (EKF). The EKF prediction and update steps are distributed among different processing units of the MAV because of their computational requirements as described in [19]. The state of the filter is composed of the position p_w^i, the attitude quaternion q_w^i, the velocity v_w^i of the IMU in the world frame, the gyroscope and accelerometer biases b_ω and b_a, and the metric scale factor λ. Additionally, the extrinsic calibration parameters describing the relative rotation q_i^s and position p_i^s between the IMU and the camera frames were also added. This yields a 24-element state vector $X = \{p_w^{i^T}\ v_w^{i^T}\ q_w^{i^T}\ b_\omega^T\ b_a^T\ \lambda\ p_i^s\ q_i^s\}$. Figure 2 depicts the setup with

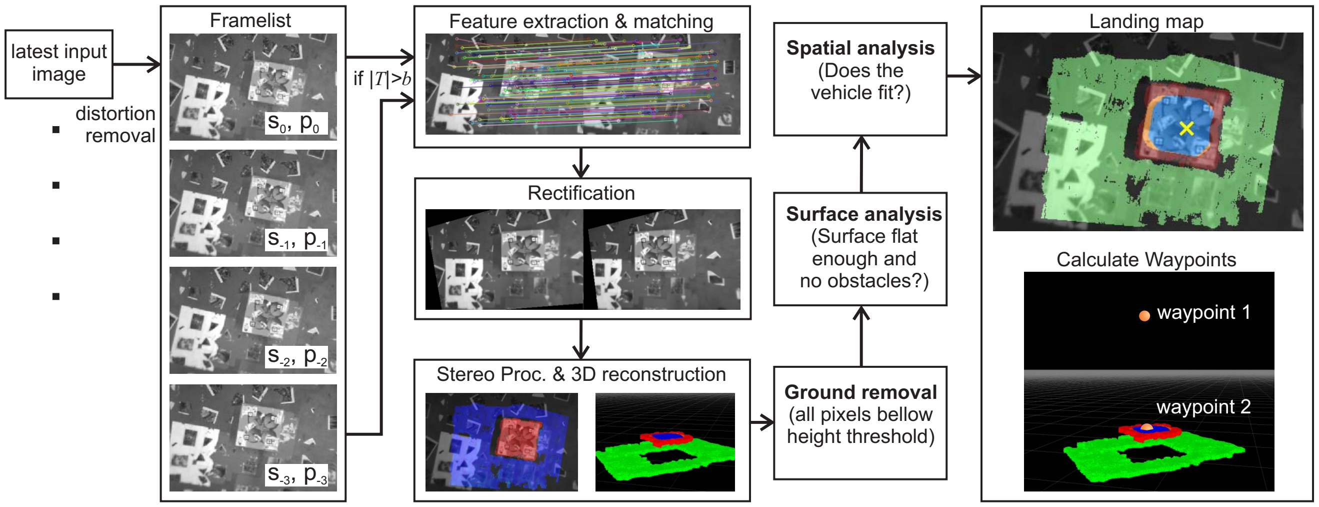
latest input image
distortion removal
Framelist
s_0, p_0
s_{-1}, p_{-1}
s_{-2}, p_{-2}
s_{-3}, p_{-3}
if $|T|>b$
Feature extraction & matching
Rectification
Stereo Proc. & 3D reconstruction
Spatial analysis (Does the vehicle fit?)
Surface analysis (Surface flat enough and no obstacles?)
Ground removal (all pixels bellow height threshold)
Landing map
Calculate Waypoints
waypoint 1
waypoint 2

Figure 3. Autonomous landing overview: New input images are stored in a frame list, together with detected features, and camera poses. For selected image pairs, camera motion is estimated, and the 3D model determined. The result of the landing site detection is a landing map which labels all pixels as: green (ground level), red (on roof top but unsafe), orange (insufficient space), or blue (safe landing area). The location with the highest confidence is labeled by $\times$ and will be approached with a two waypoint trajectory.

the IMU and camera coordinate frames and the state variables introduced above. Details about the EKF prediction and update equations for the VSLAM and IOF approach can be found in [20] and [22]. We note that for both approaches – VSLAM and IOF – roll, pitch, and the visual scale are observable. This is crucial to maintain a gravity aligned and metric navigation frame even in long-term missions.

3. Autonomous Landing

To demonstrate a high-level navigation task that uses pose information from the on-board estimation framework, we implemented an autonomous landing algorithm onboard the vehicle that analyzes images from a downward-looking camera to reconstruct the over flown scene, and detects elevated, flat surfaces, as high-vantage point landing areas. Our approach uses dense monocular 3D reconstruction and selects landing sites based on several surface criteria, such as flatness, space to fit the vehicle, and distance to obstacles.

A significant amount of work has been done on autonomous landing in unknown terrain for large helicopters using large sensor suites, such as lidar [14] or large scale stereo bars [9] that are too heavy for deployment on MAVs. Approaches that use monocular vision inputs to map and analyze terrain, often determine a surface description, such as an elevation map, from a combination of IMU and camera inputs in order to find potential landing sites. If 3D information is used, a sparse 3D reconstruction is usually done by tracking features [10, 18], estimating geometry such as planes [1] or using homographies [2] to determine a dominant planar surface for landing. Examples for methods that deploy dense monocular 3D reconstruction are [17, 12, 13] which use Bayesian and variational estimation models with known camera motion. All of them require powerful processing hardware, such as GPUs, to achieve real-time capability, and are thus not suitable for MAVs. An overview about earlier work on multi-view 3D reconstruction can be found in [15].

Our autonomous landing algorithm is especially designed to achieve reasonable short constant processing time even on limited computing hardware. It consists of three parts: 1) dense 3D reconstruction of overflown terrain by motion stereo, 2) analysis of the 3D structure in order to find potential landing site candidates, and 3) generation of a two point approach trajectory to maneuver to the best landing site.

3.1. 3D Reconstruction

Dense motion stereo is based on the same principle as conventional stereo, with the difference that the two views of a captured scene are generated by a single moving camera instead of a rigid stereo bar. Therefore, camera extrinsics (the motion between the two camera positions) have to be reconstructed for each image pair individually. Intrinsics do not change and can be estimated in advance. For selecting an image pair, we maintain a frame list of the last n camera frames (Figure 3) consisting of the rectified camera image, camera pose in world coordinates, extracted feature points in the image, and a feature track list to record features detected in subsequent frames. Image pairs for 3D reconstruction are selected using two criterias: 1) Since depth accuracy is a function of stereo baseline, we look for images that were taken at an appropriate distance apart to achieve enough depth accuracy (at ground level) which depends on the current altitude of the MAV. 2) We chose a previous image for which a minimum number of features can be tracked continuously up to the current image. After an image pair is selected, we estimate the motion (rotation R and translation t) between the two frames with a multi-planar homography alignment approach [3]. Since t can only be estimated up to scale from a monocular setup, we use the pose information from our pose estimator to scale the translation vector. With R and t, stereo rectification can be applied.

The quality of motion estimation strongly depends on the accuracy of the feature locations and, thus, is scene dependent. To discard poor motion estimates and to prevent false 3D reconstruction, we calculate the average 3D reprojection error of feature pairs and accept only image pairs with an averaged error in subpixel range. Finally, we use a real-time, block-matching stereo algorithm [5] to estimate a disparity map, from which we generate a 3D point cloud to model the captured scene beneath the MAV.

3.2. Landing Site Detection

After 3D reconstruction, the next step is to find potential landing candidates. We define the following requirements for candidate locations that are suitable for safe landing: 1) the surface around a landing site has to be planar, horizontally sloped, and free of obstacles and hazards; 2) it has to be large enough to fit the MAV; and 3) it has to provide enough free space around it for a safe approach.

To fulfill these requirements, we developed an efficient multi-step algorithm which uses the reconstructed range data to reduce the problem to a basic probabilistic model. Since our application is targeted to land on an elevated surface, we first remove all candidates close to ground level. Then we calculate the standard deviation of the disparity map along the gravity vector, as a measure of surface planarity and slope, and use this value as a landing site confidence. Standard deviation is calculated in an adaptive window which size depends on the disparity values, and thus, the MAV's altitude. In fact, we make sure that the window size (in pixels) for the standard deviation corresponds to the spacing (in meters) the MAV needs to land. Thresholding the so derived landing confidence map results in a binary landing map which labels all pixels (i.e. landing sites) as whether or not safe to land.

3.3. Approach Trajectory

The selection of a good landing site is strongly application dependent. For perch and stare applications, the MAV should land close to the edge of the elevated surface, for emergency or fast landing tasks, a safer location is preferable. In our example, we pick the landing candidate with the highest confidence and generate two approach waypoints to execute a landing maneuver, but the selection can easily be adapted to different scenarios.

For simplicity, we assume that the landing target can be reached safely by a purely vertical landing maneuver, and generate a first approach waypoint directly above the landing spot at the current MAV altitude (Figure 3). The second waypoint is the landing spot itself.

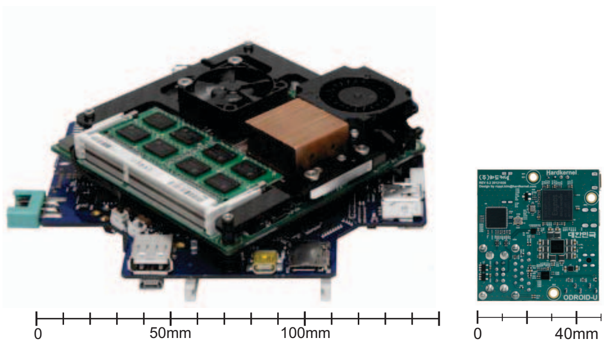

Figure 4. Mastermind flight computer (left) and Odroid-U2 (right).

4. Embedded Implementation

While the previous sections described our algorithms, this section focuses on their implementation, parameter selection, and design choices on the embedded computing architecture on the MAV.

To evaluate performance differences and the influence of weight reduction, we implemented our algorithms on three different MAV platforms: 1) an Asctec Pelican quadrotor equipped with an Asctec Mastermind flight computer (Core2Duo, 2x1.86GHz CPU, total weight: $\sim$1.3kg), 2) an Asctec Hummingbird quadrotor equipped with an Odroid-X2 flight computer (total weight: $\sim$610g), 3) an Asctec Hummingbird quadrotor equipped with a modified Odroid-U2 flight computer (total weight: $\sim$500g) (Figure 1).

Both Asctec MAV platforms use the same low-level autopilot board which includes a MEMS IMU. They are additionally equipped with a downward-looking Matrix Vision camera (mvBlueFOX-MLC200wG, CMOS, 752×480, grayscale, global shutter, max. 90fps, 18.3g with 100 degree FOV lens) that is connected to the flight computer. The Odroid boards (manufactured by Hardkernel) are based on the Samsung Exynos 4412 System-on-a-Chip (SoC), which includes a quad core microcontroller for mobile applications that provides four ARM-cortex A9 for parallel computation, while only consuming 2.2W (CPU only). The Odroid-X2 is a development board which served for initial implementation and testing, while the Odroid-U2 is our final target platform. For our implementation, we removed all non-necessary components from the U2 resulting in a total weight of 12g. Details about the different platforms can be found in Table 1, whereas Figure 4 illustrates the size difference between the Asctec Mastermind and the Odroid U2.

Figure 5 gives an overview of the distributed implementation of our approach on the vehicle. All computational expensive components are executed on the high level flight computer, including VSLAM, IOF, and pose filter update (EKF-update) as well as landing site detection. The EKF-update is passed down to the prediction loop that is executed on the autopilot board for efficiency reasons. The prediction loop, which includes IMU integration, and the position controller that uses the estimated pose to control the vehicle, both run at 1kHz on a dedicated ARM7 microcontroller.

Table 1. SWaP and performance of tested computing platforms executing VSLAM pose estimation.

Platform	Size (footprint)	Weight	Power consumption	Cores	Vision front-end frame rate (Hz)	CPU load	Workload
Asctec Mastermind	144x135mm	300g	30W	2	30	59%	30%
Odroid-X2 dev. board (4412) including heat sink	90x94mm	122g	8W	4	30	125%	31%
Odroid-U2 (4412) stripped down version	48x52mm	12g	5W	4	30	125%	31%

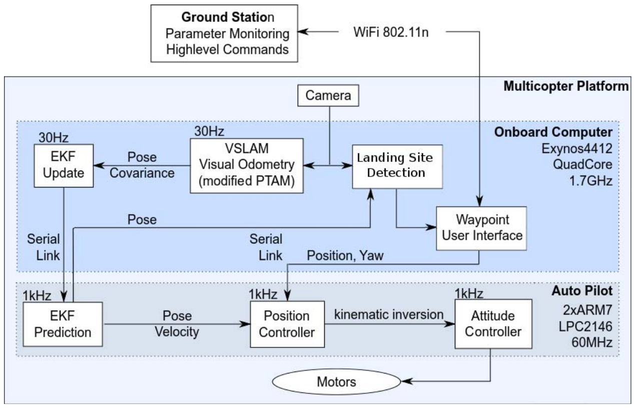

Figure 5. System overview of the vision-aided pose estimation frame work.

We ported the initial estimator implementation from the Asctec Mastermind to the 4412 boards by using system specific changes in order to speed up the execution on the SoCs. We used a highly ARM-customized Ubuntu version as operating system and Robot Operating System (ROS) for inter process communication. Since both 4412 boards use the same CPU but different interface hardware, the operation system setup reflected the different hardware setup, but the implementation of our navigation software was basically identical. The straight forward port of the full pose estimation frame work without making use of parallelism yielded a frame rate of 10Hz for vision updates of the VSLAM system (EKF update steps were nearly negligible in all implementations). For optimization, we adapted the VSLAM code to use vectorized NEON instructions for hardware acceleration, and developed a core balancing module to distribute individual processes among different cores. Our VSLAM implementation primarily consist of a *Tracking* and a *Mapping* part which we enforce to be executed on separate cores. *Tracking* is the most critical part, since it yields instantaneous pose measurements which are used to generate filter updates. Therefore, running this part on a dedicated core ensures uninterrupted pose handling at all time. *Mapping* is responsible for pose refinement and windowed bundle adjustment, and is thus less time critical. Note, that the adjustments are refinements and we do not use global loop closure techniques, which avoids large and abrupt pose changes. Since the mapping task runs at a lower frequency and is less time critical, it shares its dedicated core with other system tasks. After optimization, the VSLAM vision front-end produced visual pose estimates at a stable 50Hz rate. Our IOF approach was already optimized to run at 50Hz on a single core 1.6GHz Intel ATOM processor. ARM specific adaptations and NEON instructions yielded the same performance on a single core on the 4412. Finally, the EKF update part of the pose filter software was adapted to run on the 4412, which completed the software port. The landing framework was initially implemented on a standard PC and then ported to the 4412. Software optimization included NEON adaptations and the reduction of image resolution to 376×240 which increased the processing speed significantly, while still maintaining sufficient resolution. In fact, our approach compensates for smaller depth resolutions by automatically choosing larger baselines between selected image frames.

4.1. Performance Comparisons

For performance comparison, we executed our VSLAM-based visual inertial state estimation framework (vision front-end and EKF state estimation) on the three different architectures described in Section 4. The camera frame rate was reduced to 30Hz on all architectures, since previous experiments suggest that EKF updates at this rate are sufficient for agile quadcopter navigation when fused with high rate (1kHz) IMU data [19]. At this new frame rate, the CPU load on the Core2Duo was 59% of the overall available 200% (2 cores $\times$ 100%) translating to an overall system workload of 30%. Running our software at the same frame rate on the 4412 X2 and U2 resulted in 125% CPU load out of the available 400% (4 cores $\times$ 100%), translating to a 31% system load. This demonstrates, that we have similar system reserves available on both architectures (Core2Duo and 4412) for higher level autonomous navigation tasks, while weight was reduced by 96% and power consumption was reduced by 74% (Table 1).

When running the landing site detection algorithm in parallel with our pose estimation frame work on the Hummingbird/U2, we achieved a frame rate of 1Hz for landing map updates. Our experiments showed that this frame rate is reasonable for fully autonomous landing site detection.

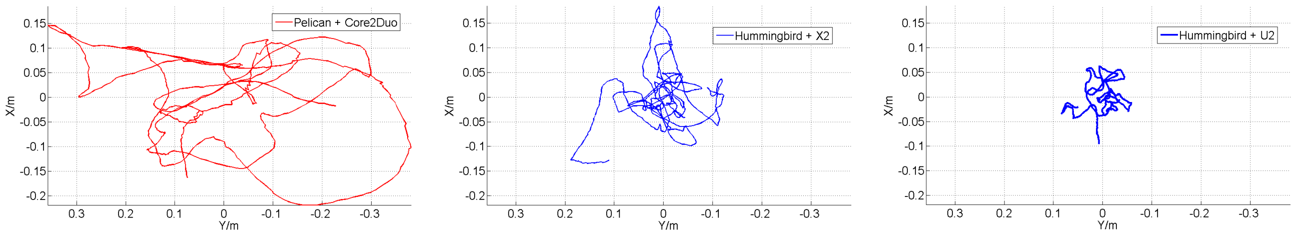

Figure 6. Hover performance of the Pelican (left), the Hummingbird with X2 (middle), the Hummingbird with modified U2 (right).

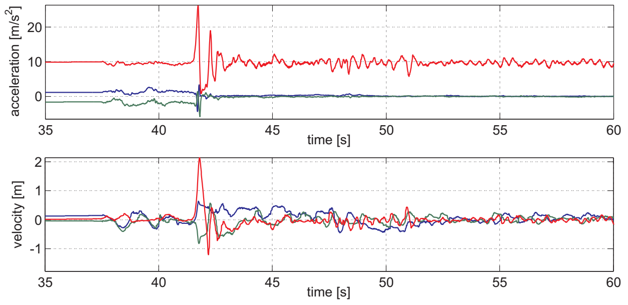

Figure 7. Throw-and-go launch: Acceleration (top) and velocity (bottom) in x (blue), y (green), and z (red) direction when throwing the MAV in the air. At the throw, the MAV experiences excitations of up to 16.5m/s^2 and 190°/s. The velocity rises up to 2.3m/s. The quick stabilization after about 4s allows for subsequent VSLAM initialization.

4.2. Experimental Evaluation

To evaluate the influence of the reduced weight on the control stability of the platform, we executed a position hold maneuver with all three vehicle/flight computer configurations, where the MAV was controlled only with position estimates from our VSLAM pose estimation software (again executed at a frame rate of 30Hz).

Neglecting the influence of different flight performances of the two quadrotor systems, the reduced gross weight resulted in significantly better control performance: the hovering ellipse was reduced from ±35cm for the heavy Asctec Pelican with Mastermind (RMS(x y z)=[8.3cm 15.8cm 1.5cm]) to about ±15cm for the Hummingbird with the X2 (RMS(x y z)=[5.4cm 5.7cm 1cm]) and to ±7cm for the Hummingbird with the final stripped down version of the U2 (RMS(x y z)=[2.9cm 3.0cm 0.8cm]) (Figure 6).

Performance evaluation of our IOF approach showed that it is sufficiently robust to estimate the vehicle pose even in drastic motion as it occurs when tossing the MAV in the air. This and the in-built self-calibrating capability of the filter framework ensure quick MAV deployment. In Figure 7, we start our IOF based state estimation at t=38s while holding the vehicle in hand. The vehicle is tossed in the air at t=42s. The initial period of 4 seconds is already sufficient for the states to converge in the pose filter in order to stabilize the MAV after a throw. About 1 second after the throw, the vehicle stabilizes already in attitude and in velocity. The convergence of the scene depth requires about 6 seconds longer. This is due to a non-optimal initialization of the metric scale factor which generally converges slower than the other states in the system. Once all states are converged and the vehicle fully stabilized (after about 7 seconds) the system is ready to initialize the VSLAM system.

Our autonomous landing experiments consisted of landing site detection, and target approach. All experiments were conducted with the Asctec Hummingbird carrying the modified U2 flight computer, and running the pose estimation frame work and landing site detection algorithm in parallel. We conducted several indoor experiments were the

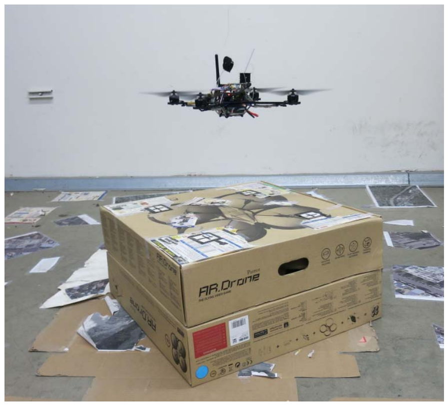

Figure 8. Asctec Hummingbird with U2 landing on elevated surface.

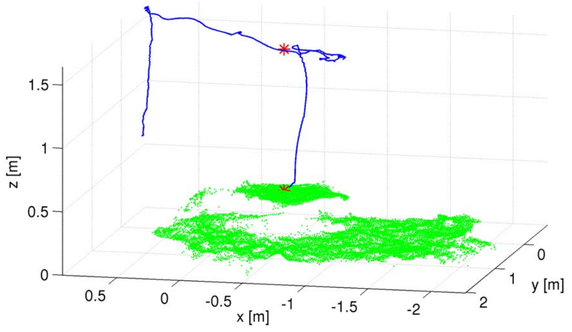

Figure 9. Indoor landing experiment: The MAV is commanded to take off, rise to 1.5m altitude, and autonomously fly towards the landing platform (trajectory in blue). Once a valid landing spot is detected, 2 approach waypoints are generated (stars) for the landing maneuver. The tolerance radius on the first point is larger than on the landing point to allow high precision landing.

landing target consisted of a cardboard box to simulate an elevated landing surface, such as a rooftop (Figure 8). The quadrotor was commanded to fly over the elevated surface by defining manual waypoints, which were approached by the vehicle autonomously while executing the landing site detection algorithm to analyze the area beneath the MAV. As soon as an appropriate landing spot was detected, the two approach waypoints were submitted and executed by the vehicle. Figure 9 illustrates the reconstructed 3D point cloud of the box and ground surface together with the flight trajectory and the issued approach waypoints. The vehicle took off to the left of the depicted scene and landed correctly on top of the box. Example scene views, together with a resulting landing map are illustrated in Figure 3. All pixels located in the middle of the box have been labeled correctly as safe to land (blue), whereas pixels close to the edges of the box are detected as either unsafe (red) or provide not enough space for landing (orange).

5. Conclusion

This paper introduces the currently smallest micro helicopter ($\sim$500g) navigating purely vision based and with passive sensing only. With the implementation of our pose estimation frame work on the Exynos 4412 based Odroid-U2 single board computer, we created an ultra-light weight flight computer module that provides a MAV platform with reliable pose estimates in GPS-denied environments at high frame rate, and executes high-level navigation tasks such as autonomous landing on an elevated platform. The system performs continuous self-calibration and is able to be deployed by tossing it into the air, which extends the platform from a *power-on-and-go* system to a *throw-and-go* MAV.

The implementation on an ultra-light weight platform of only 12g is a huge step towards ultimately having a fully capable avionics package (flight computer, camera, and IMU) under 15g. It will enable fully autonomous control of ultra-small quadrotor systems (as e.g. the 15cm, 25g Bitcraze miniature quadrotor system [4]) that can be deployed for indoor and outdoor ISR missions in confined spaces while maintaining stealth.

References

[1] S. Bosch, S. Lacroix, and F. Caballero. Autonomous detection of safe landing areas for an uav from monocular images. In *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 5522–5527, 2006.

[2] R. Brockers, S. Susca, D. Zhu, and L. Matthies. Fully self-contained vision-aided navigation and landing of a micro air vehicle independent from external sensor inputs. *Proc. SPIE*, 8387:83870Q–1 – 83870Q–10, 2012.

[3] Y. Cheng. Real-time surface slope estimation by homography alignment for spacecraft safe landing. In *Proc. IEEE Intl. Conf. on Robot. and Autom.*, pages 2280–2286, 2010.

[4] Crazyflie micro quadrotor: http://www.bitcraze.se/crazyflie/.

[5] S. Goldberg, M. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *IEEE Aerospace Conf. Proc.*, pages 2025–2036, 2002.

[6] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research (IJRR)*, 30(1):56–79, 2011.

[7] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. Intl. Sym. on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007.

[8] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A constant time efficient stereo SLAM system. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[9] M. Meingast, C. Geyer, and S. Sastry. Vision based terrain recovery for landing unmanned aerial vehicles. In *Proc. IEEE Conf. on Decis. and Contr.*, pages 1670–1675, 2004.

[10] J. Montgomery, A. Johnson, S. Roumeliotis, and L. Matthies. The jet propulsion laboratory autonomous helicopter testbed: A platform for planetary exploration technology research and development. *J. Field Robot.*, 23(3-4):245–267, 2006.

[11] A. Mourikis and S. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proc. IEEE Intl. Conf. on Robot. and Autom.*, 2007.

[12] R. A. Newcombe, J. S. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time, 2011.

[13] M. Pizzoli, C. Forster, and D. Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *Proc. IEEE Intl. Conf. on Robot. and Autom.*, 2014.

[14] S. Scherer, L. J. Chamberlain, and S. Singh. Autonomous landing at unprepared sites by a full-scale helicopter. *Robotics and Autonomous Systems*, 2012.

[15] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE Intl. Conf. on Comp. Vis. and Pat. Recogn.*, pages 519–528, 2006.

[16] S. Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *Proc. IEEE Intl. Conf. on Robot. and Autom.*, 2011.

[17] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Proc. 32nd DAGM Conference on Pattern Recognition*, pages 11–20, 2010.

[18] T. Templeton, D. H. Shim, C. Geyer, and S. S. Sastry. Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft. In *Proc. IEEE Intl. Conf. on Robot. and Autom.*, pages 1349–1356, 2007.

[19] S. Weiss, M. W. Achtelik, M. Chli, and R. Siegwart. Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV. In *Proc. IEEE Intl. Conf. on Robot. and Autom.*, 2012.

[20] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart. Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *J. Field Robot.*, 30(5):803–831, Aug. 2013.

[21] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *Proc. IEEE Intl. Conf. on Robot. and Autom.*, 2012.

[22] S. Weiss, R. Brockers, and L. Matthies. 4dof drift free navigation using inertial cues and optical flow. In *Proc. IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, pages 4180–4186, 2013.