International Micro Air Vehicle Conference and Flight Competition (IMAV2013)
17-20 September 2013, Toulouse, France

3D Perception for Autonomous Navigation of a Low-Cost MAV using Minimal Landmarks

Benjamin Ranft[1], Jean-Luc Dugelay[2] and Ludovic Apvrille[3]

[1] FZI Research Center for Information Technology, Karlsruhe, Germany
ranft@fzi.de
[2] Institut Eurécom, Sophia Antipolis, France
dugelay@eurecom.fr
[3] Institut Mines-Télécom / Télécom ParisTech, Sophia Antipolis, France
ludovic.apvrille@telecom-paristech.fr

Abstract

We present an implementation of autonomous navigation for Micro Air Vehicles which is well-suited for inexpensive models: It only relies on a single camera and few on-board sensors to solve the challenges of flight planning and collision avoidance. Artificial landmarks are merely required in places with ambiguous further flight paths, e. g. corridor crossings or junctions. There they provide topological localization and enable our system to perform tasks like way point following.

Even without any direct 3D sensor, our system is able to reconstruct metric distances from its monocular camera using two complementary methods: An oscillating motion pattern is superimposed to regular flight to reliably estimate up-to-date 3D positions of sparse image features. As an alternative, a specific flight maneuver can virtually create a vertical stereo camera to provide depth information densely across most pixels at single points in time. The unknown metric scale inherent in employing a single camera is determined by evaluating further sensors via a robust two-stage approach. We use the reconstructed 3D scenes to traverse free space and avoid obstacles.

1 Introduction

The popularity, availability and range of applications of Micro Air Vehicles (MAV) – especially quadcopters – has been steadily increasing over the last few years. While the mechanical performance of rotor-driven models has long been satisfactory, enabling applications like localization, mapping and autonomous flight using minimal sensors and infrastructure still presents various challenges to research: Especially for MAVs, on-board sensors as well as processors should be inexpensive, but also lightweight and energy-efficient. Implementations therefore must be able to cope with limited and noisy inputs. At the same time, either their complexity needs to be feasible for on-board processing or their robustness against signal latencies and interruptions has to allow for remote operation.

Figure 1: *AR.Drone 2.0* quadcopter: Its 4 degrees of freedom during flight are indicated by arrows.

As an example of such applications, we present a complete system for fully-autonomous indoor navigation based on the monocular forward-facing camera and supplementary on-board sensors of the inexpensive *Parrot AR.Drone 2.0* quadcopter [17] shown in fig. 1. Regarding infrastructure, our system requires a standard PC for remote processing and control, and landmarks for topological

localization at crossings or junctions, but no external cameras or radio beacons. Our respective labs productively employ the presented system i. a. for studying and mitigating the quadcopter's impact on privacy while guiding visitors in an office environment, as well as for teaching computer vision and control theory. A preliminary demonstration video is available at [2].

The remainder of this paper is organized as follows: Section 2 introduces related works and differentiates them from our contributions. After section 3 has given an overview of our system and its fundamentals, the two complementary methods for sparse and dense 3D reconstruction as well as for landmark-based navigation are described in sections 4 to 6, including respective experimental results. Section 7 concludes the paper and presents an outlook on future works.

2 Related Work and Contribution

MAV research connects a diverse range of topics such as control systems, computer vision, sensor fusion and artificial intelligence. Several publications are related to individual aspects of this paper: [18] avoids collisions based on re-projected sparse feature correspondences between images from a single camera. This technique's inherent ambiguity of absolute scale can be resolved via integrating additional sensors – exemplarily, inertial measurement units (IMU) are employed by [1], [6] and [15] to enable drift-free localization, precise figure-flying and autonomous mapping respectively. We extend this approach in two ways: Firstly, a preferred method to estimate the absolute distances of re-projected sparse 3D points evaluates the quadcopter's position relative to the ground plane. It is therefore less susceptible to noise in IMU measurements. Secondly, we superimpose a circular motion pattern in vertical and lateral direction to regular flight so that feature correspondences can be re-projected more precisely and reliably.

Besides the above structure from motion-based approaches, various sensors to directly measure a MAV's 3D environment are available, including time-of-flight and structured light cameras or LIDAR. As another example, [7] primarily uses on-board stereo cameras to explore and map unknown environments. We propose to emulate such a sensor by evaluating monocular images before and after a change in flight altitude. While saving one camera's weight and power consumption, this approach also introduces algorithmic challenges and inherent limitations.

The more complex quadcopter models used in some of the works above allow performing at least parts of the respective computations on-board. In our system, processing is also distributed among a remote PC and the *AR.Drone* itself, but the latter can only run manufacturer-supplied code such as longitudinal and lateral velocity tracking. Nevertheless, this does not necessarily limit our system's range – e. g. a six-legged ground robot has been demonstrated to be a suitable base station for cooperative terrain exploration with a very similar quadcopter model [10].

3 System Overview and Fundamentals

As mentioned initially and shown in fig. 2, our quadcopter is complemented by a remote PC for performing custom tasks like 3D reconstruction, landmark recognition and flight control. The following paragraphs will introduce each of the depicted modules:

Our MAV of choice is the *Parrot AR.Drone 2.0*, whose standard quadcopter design enables flight with four degrees of freedom: Vertical motion and yaw rotation are independent, while forward/backward and sideways motion is achieved via pitch and roll rotation respectively. Although being marketed mainly as a toy for private use at a price of $ 300, it provides various sensors, and is safe to use and robust. In contrast to more expensive models, custom code cannot easily be run on-board, and hardware extensions are limited to vendor-supported devices – as of 2013, these only include USB flash drives for recording flights and an optional GPS receiver.

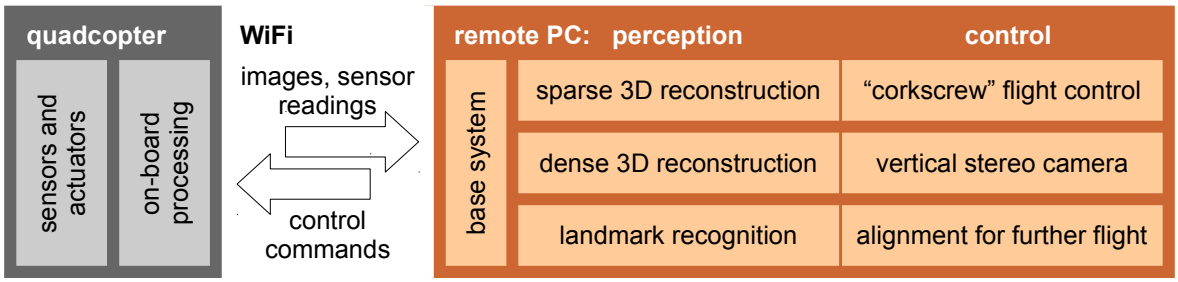

Figure 2: Overview of our system's modules: The perception methods may be run concurrently, but only one of their associated control strategies is selected based on their respective results.

Among the aforementioned on-board sensors are a 3-axis accelerometer, both an ultrasonic and a pressure-based altimeter, but also a gyroscope and a magnetometer for measuring the quadcopter's angular velocities and absolute orientation. Their readings are combined with the optical flow determined from a downward camera to estimate the quadcopter's translational velocities at short intervals and with reduced drift [3]. Another forward-facing camera provides images with a resolution of 1280×720 pixels for remote processing at a rate of up to 30 Hz. However, in order to transfer the video stream more steadily and reduce compression artifacts, we limit both to 640×360 px and 15 Hz respectively.

The remote PC connects to the quadcopter via WiFi. At this, our base system employs *Robot Operating System* [16], a widely-used open-source middleware. A supplementary driver [14] conveniently provides the *AR.Drone*'s camera images and sensor measurements, and accepts normalized control commands $\in [-1, 1]$ for each of the quadcopter's four degrees of freedom.

Because our methods for environmental perception generate metric deviations between the quadcopter's current and target 3D position and yaw angle, we use discrete-time PID controllers to convert them to the required normalized commands. Table 1 shows their respective parameters, which have been determined via the closed-loop Ziegler-Nichols method: In individual experiments, proportional-only control is used to hover at a certain altitude or yaw angle, or exactly above a longitudinal or lateral line on the ground. The controller's gain is increased until, at its ultimate value k_U, the quadcopter performs a permanent marginally stable oscillation with period T_U. Both values are used in a heuristic rule [13] for finding PID control parameters which achieve quick settling without overshoot. As the controllers' outputs may well exceed the $[-1, 1]$ interval and therefore the quadcopter's capabilities, an anti-windup logic prevents their error integrals from building up and – once the set point has been reached – causing significant overshoot while being unwound again.

One last part of our base system is image undistortion – it is required since all our methods for environmental perception expect the pinhole camera model to be valid. This model greatly simplifies the re-projection of image points into 3D, and approximately holds for the *AR.Drone*'s downward camera. Its forward-facing camera however shows significant barrel distortion, which can be described and corrected using the *Brown-Conrady* model [4]. The undistorted images cover a $64\,^\circ$ horizontal field of view on 736×360 pixels. Undistortion is our system's only task to exploit data-parallelism

degree of freedom	k_U	T_U	$k_P = k_U / 5$	$k_I = 2\,k_P / T_U$	$k_D = k_P\,T_U / 3$
longitudinal and lateral	1.2	4.65	0.24	0.10	0.37
vertical	2.9	2.50	0.58	0.46	0.48
yaw-angular	5.4	0.95	1.08	2.27	0.34

Table 1: PID control parameters for each of the quadcopter's degrees of freedom: k_U and T_U are determined experimentally and used to find the actual parameters via a heuristic rule [13].

through multiple CPU cores or optionally a GPU in order to minimize latency. All further processing runs sufficiently fast even on an *Intel Core 2 Duo T7700*, although we selected efficient algorithms and implementations mainly to maintain the possibility of using the embedded on-board processor of a different quadcopter model in the future.

To conclude this overview, we will briefly introduce the three combinations of environmental perception methods and flight control strategies listed in fig. 2 – their respective implementations and experimental results on the other hand will be described in individual sections later on.

- Sparse 3D reconstruction may be used continuously during regular flight and therefore is our preferred method of perception. It usually yields the spatial locations of few hundreds of distinct image points. Their accuracy largely depends on the quadcopter's motion: Vertical and sideways movements are particularly beneficial, which is why our associated control strategy superimposes an oscillation in those directions, hereby creating a corkscrew-shaped flight trajectory.
- Dense 3D reconstruction can alternatively provide an estimated distance for most of the 265.000 pixels of an image, but in return requires exclusive flight control to virtually create a vertical stereo camera through a change in altitude. Because regular flight needs to be interrupted for this maneuver, results are dense in space yet sparse in time.
- Landmarks to be detected and recognized by our system are only required in places offering multiple possible directions for further flight. There they provide topological localization of the quadcopter within the set of marked crossings, including its inbound direction. Using this information, it can assume a previously defined position and outbound direction relative to the landmark before resuming 3D reconstruction-based flight.

While undistorted input images and supplementary sensor data are processed in parallel by all perception methods, only one control strategy's outputs must be selected for being sent to the quadcopter. We prioritize the strategies based on their internal state: Once triggered, the altitude change maneuver for dense 3D reconstruction is never interrupted. Otherwise, landmark-based navigation is in effect while a landmark has been detected and the corresponding further flight orientation has not yet been assumed. Lastly, continuous sparse 3D reconstruction and "corkscrew" flight are intermitted with dense 3D scans on demand.

4 Sparse 3D Reconstruction and "Corkscrew" Flight

The basis of this method of environmental perception is formed by sparse optical flow, i. e. the change in pixel coordinates of corresponding distinct points from two subsequent images. Our system usually evaluates directly consecutive frames from the quadcopter's forward camera, but may make one exception in case of very slow flight: Because a sufficient translational movement between both views is required to obtain accurately reconstructed 3D points, a previous image may be kept and processed with a series of new images until this movement has built up.

The process of estimating this translational as well as the rotational movement of a camera is called visual odometry – a real-time open-source implementation is offered i. a. by *LIBVISO2* [8] for both stereoscopic and monocular cameras. Because the latter version is particularly aimed at ground vehicles however, we extended it for use in conjunction with a quadcopter. The following list outlines the whole process of obtaining 3D points from an image pair – state-of-the-art algorithms from [9] are briefly summarized while our extensions are described in detail:

1. To efficiently find correspondences between images, *LIBVISO2* uses custom implementations: Its feature detector distinguishes between 4 classes. Its descriptor is neither scale- nor rotation-invariant, but quickly compute- and comparable. The matching strategy considers intra-class matches only and uses the results from an initial subset of points as priors for the remaining set.

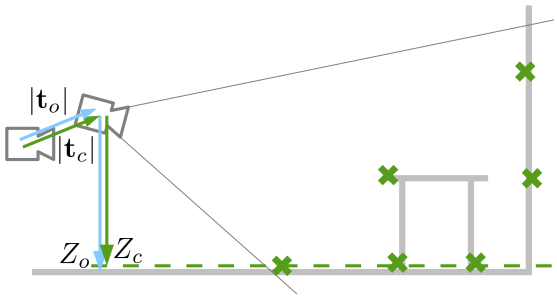

Figure 3: The ground plane is correctly identified by the 4th step of sparse 3D reconstruction. The camera's height Z_c above it may safely be assumed to equal the altitude Z_o measured by the on-board ultrasonic sensor.

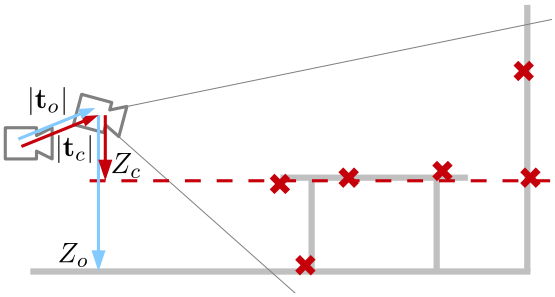

Figure 4: The tabletop is mistaken for the ground plane. The 3D points' distances and the camera's motion $\mathbf{t}_c$ are over-estimated by a common factor, which can be corrected by incorporating $\mathbf{t}_o$ from on-board odometry.

2. The rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}_c$ between both camera poses are determined as intermediate results[1]: As a basis, applying the normalized eight-point algorithm in a RANSAC scheme yields a robust estimate of the fundamental matrix $\mathbf{F}$, which relates the two views. The camera matrix $\mathbf{K}$ is used for converting $\mathbf{F}$ to the essential matrix $\mathbf{E}$, from which four potential solutions for the camera's motion can be constructed. However, each one's translation vector is only defined up to scale – its metric length remains to be determined.

$$\mathbf{E} = \mathbf{K}^\top \mathbf{F} \mathbf{K} = [\mathbf{t}]_\times \mathbf{R} \tag{1}$$

3. Given the camera's motion, a 3D position $\mathbf{X}_i = [X_i\, Y_i\, Z_i]^\top$ of each point correspondence i can be reconstructed very efficiently via homogeneous linear triangulation, which merely requires the singular value decomposition of a 4×4 matrix. The correct tuple $(\mathbf{R}, \mathbf{t}_c)$ among the four potential solutions from the previous step can be easily determined because ideally all 3D points are reconstructed in front of both cameras for one of them only. However, since $\mathbf{t}$ is still only determined up to scale, the same holds true for $\mathbf{X}_i$.

4. Our preferred method of estimating the above common scale of all $\mathbf{X}_i$ and $\mathbf{t}$ relies on the assumption that – as shown in fig. 3 – a horizontal ground plane extends from below the quadcopter into its camera's field of view. This plane's position w. r. t. the quadcopter can be described in terms of flight altitude Z_o, pitch θ and roll ψ, all of which are measured directly and accurately on-board. Using these angles, each 3D point $\mathbf{X}_i$ can be projected to the ground plane's normal vector:

$$Z_i = \mathbf{n}^\top \mathbf{X}_i = \begin{pmatrix} \cos(\theta)\sin(\psi) \\ \cos(\theta)\cos(\psi) \\ -\sin(\theta) \end{pmatrix}^\top \mathbf{X}_i \tag{2}$$

   After finding the distance Z_c of the largest cluster within all Z_i below the quadcopter, the scale factor s is determined so that Z_c matches the altitude Z_o measured by the ultrasonic sensor, i. e. $s = Z_o/Z_c$. This scale is then applied to each 3D point $\mathbf{X}_i$ and the translation vector $\mathbf{t}_c$.

5. An optional second stage of scale estimation more ordinarily incorporates the translation vector $\mathbf{t}_o$ obtained by integrating measurements from the on-board accelerometer. Since our quad-

[1] Algorithms such as bundle adjustment, which estimate the camera's motion and the feature correspondences' 3D positions jointly rather than successively, may yield more accurate results [9], but are considerably more computationally demanding than the presented approach.

Figure 5: Sparse 3D reconstructions: Blue/purple lines show optical flow vectors consistent/ conflicting with the camera's motion. The points' color represents their longitudinal distance – red indicates 1 m and below, cyan for 10 m and above. A larger circle marks the target flight direction.

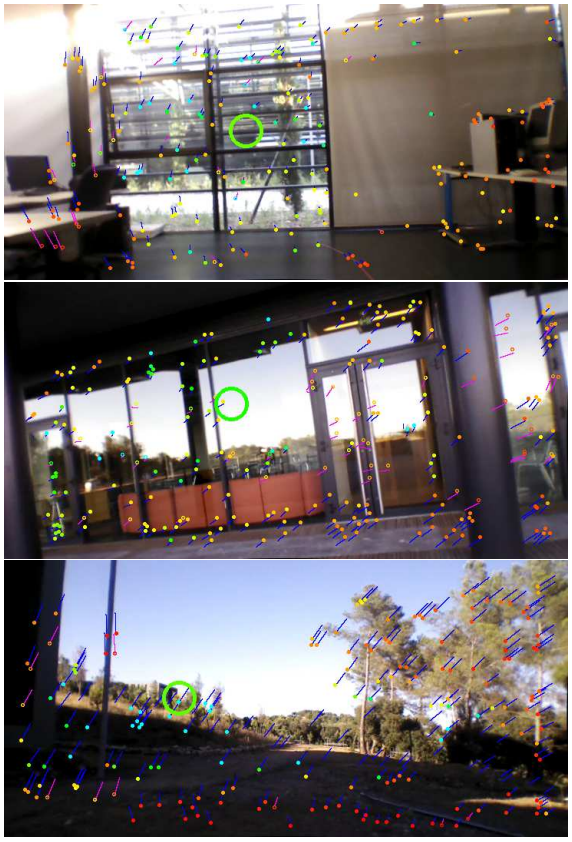

Figure 6: Imperfect sparse 3D reconstructions: A path through a window is planned because of too few correspondences on its blinds (top). Mostly mirrored points are reconstructed (middle). Difficult lighting caused erroneous estimates of camera motion and 3D point distances (bottom).

copter's downward camera compensates them for drift, the scale factor can similarly be determined as $s' = |\mathbf{t}_o| \,/\, |\mathbf{t}_c|$. Nevertheless, the above method has proven to be more precise if the ground actually is horizontal and correctly identified within the set of 3D points. Therefore, s' should only be applied if that assumption does not hold, e. g. above stairs or – as illustrated in fig. 4 – if tabletops predominate floor within the field of view. Such situations can be recognized by large deviations between $\mathbf{t}_c$ and $\mathbf{t}_o$: Our implementation requires s' to change the scale by at least 20% in order to become effective.

Figs. 5 and 6 show exemplary results of sparse 3D reconstruction as well as marked target flight directions. Our approach for determining the latter is similar to [18]'s virtual *directional distance sensors*: We equally divide the camera's field of view into a $(j = 3) \times (k = 5)$ grid of such sensors, each of which provides a single distance measurement d_{jk}. At this, rather than finding the closest cluster among the distances $d_i = |\mathbf{X}_i|$ within a sensor's region of interest, the 25^{th} percentile distance yields comparable results at a reduced computational complexity. We first determine the target flight

direction in image coordinates $\mathbf{x}_t$ by computing a weighted mean of each virtual distance sensor's center $\mathbf{x}_{jk}$:

$$\mathbf{x}_t = \begin{pmatrix} u_t \\ v_t \end{pmatrix} = \frac{\sum\limits_{\substack{j=1..3 \\ k=1..5}} \mathbf{x}_{jk}\, d_{jk}^2}{\sum\limits_{\substack{j=1..3 \\ k=1..5}} d_{jk}^2} \tag{3}$$

However, a grid cell may potentially lack any feature matches and 3D points, e. g. in case it merely views a blank wall. Our implementation then artificially sets the associated distance to zero in order to prevent the quadcopter from flying towards such uncharted regions.

The target flight direction only affects the quadcopter's yaw rotation and forward velocity, since its sideways and vertical degrees of freedom are reserved for the aforementioned oscillations. A conflict of goals must be resolved when defining their amplitude and frequency: Higher values ensure sufficient motion for precise 3D reconstruction, but require more battery power and a larger clearance space. We found 0.3 m and 0.33 Hz respectively to be a good compromise in an office environment. Because of this scenario we also decided to currently ignore the target direction's vertical coordinate v_t: The quadcopter should fly around tables rather than passing above or below them. The horizontal coordinate can be converted into an angular deviation $\Delta\varphi = \arctan\left(\left(u_t - c_u\right)/f_u\right)$, which is not only used in closed-loop PID yaw rate control, but also in a heuristic for determining the forward flight velocity: It is highest if the quadcopter is well-aligned with the target direction, while angular deviations of $\pm 32\,^\circ$, i. e. the edges of its field of view, lead to in-place rotations without any forward component.

5 Dense 3D Reconstruction with a Virtual Stereo Camera

In contrast to the previous method, a stereo camera allows estimating a distance Z_i for the majority of pixels. Our system can emulate such a sensor by moving the quadcopter perpendicularly to its camera's optical axis and analyzing images buffered before and after that maneuver. Hereby, virtually creating a vertical stereo camera through an in-place change in flight altitude has several advantages over the default horizontal arrangement: The *AR.Drone*'s ultrasonic height sensor allows to precisely measure the baseline distance between both views and to adapt it to the current scene: 0.20 m have proven suitable for the indoor scenarios presented in fig. 9, but larger values likely yield more precise outdoor results. We also found a vertical instead of a lateral offset to be more quickly and smoothly controllable using on-board odometry – due to the maneuver's duration of only 0.5 s, the update rate of visual odometry is insufficient here. Finally, as also shown in fig. 9, the field of view of a vertical stereo camera's results is also limited vertically, which is the less relevant dimension for our application.

Efficient generic implementations of dense stereo matching such as [11] require input image pairs to be rectified: Only if all corresponding points are ideally located on the same column, finding their disparity $d_i = v_{i,after} - v_{i,before} = |\mathbf{t}|\, f/Z_i$ becomes a valid 1D optimization problem. As sketched

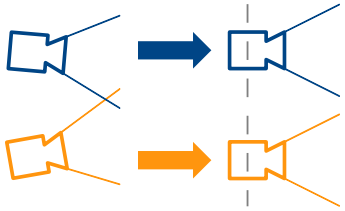

Figure 7: Rectification of one image each before and after a height change, based on camera motion estimate according to section 4

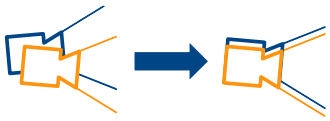

Figure 8: Video stabilization for moving object masking, used while hovering before or after a height change maneuver for dense 3D perception

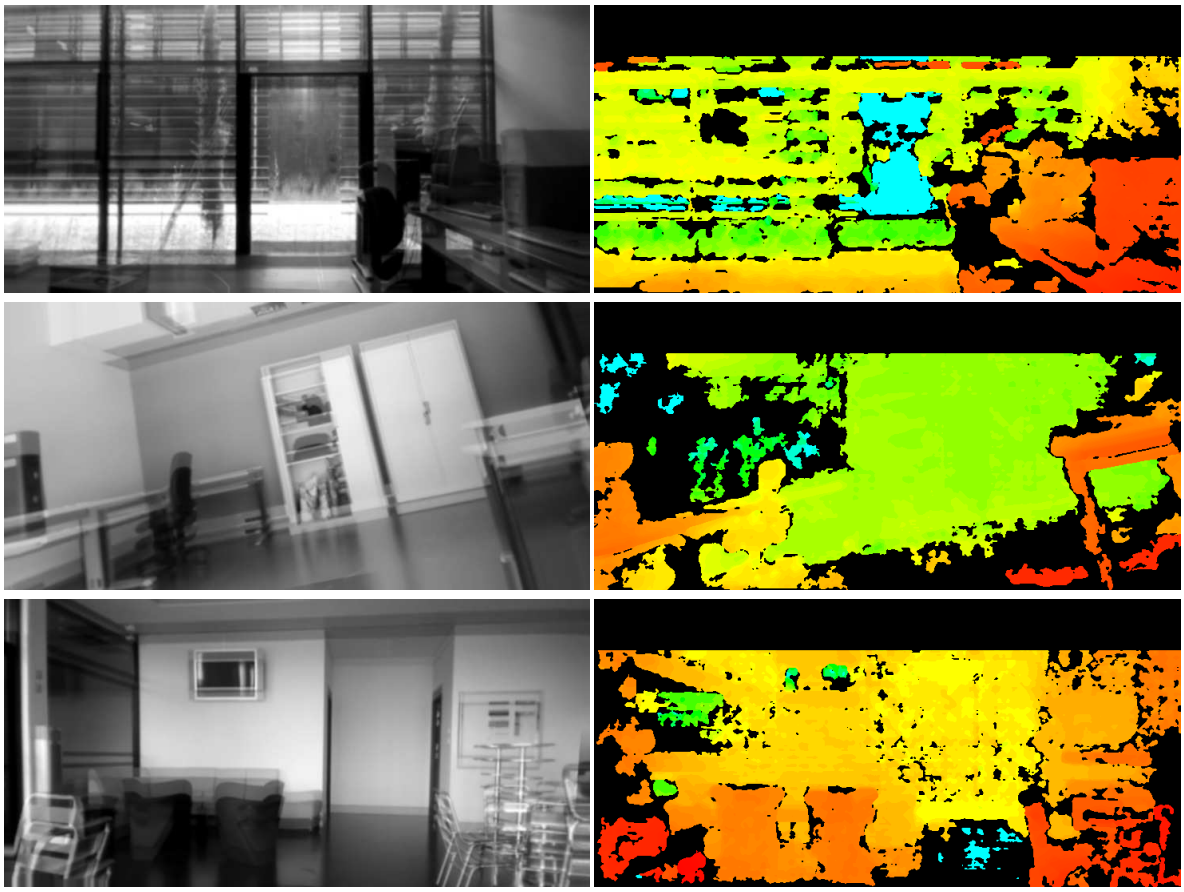

Figure 9: Dense 3D reconstruction results: The overlayed rectified images before and after the height change visualize the precision of the estimated camera motion (left). Therefore, any standard implementation for distance reconstruction, e. g. [11], may be used without modification (right).

in fig. 7, this is achieved by applying perspective transformations to both views such that their image planes and v-axes coincide. The associated homographies can be computed from the camera matrix $\mathbf{K}$ and motion $(\mathbf{R}, \mathbf{t}_c)$. While the former can be statically calibrated, our system requires precise estimates of the latter for each individual pair of images from before and after a change in altitude. Nevertheless, they can successfully be provided by applying the steps described in section 4 directly to both images. Rectification may however fail if wind or another disturbance cause excessive lateral or especially longitudinal motion: We detect such cases within the camera-based translation estimate $\mathbf{t}_c = [t_{cx}\, t_{cy}\, t_{cz}]^\top$ by evaluating $4t_{cx}^2 + 16t_{cz}^2 > t_{cy}^2$ and – if true – repeating the maneuver.

One conceptual disadvantage of the proposed method can merely be mitigated algorithmically: While an actual stereo camera captures both images simultaneously, our system allows objects to move during the 0.5 s height change. Therefore, they often do not meet the above same-column criterion and cause incorrect distance estimates. Our preliminary approach can at least mask the potentially affected results before further processing: We detect moving objects by hovering in place for 0.75 s before and after the maneuver. As sketched in fig. 8, the first and last image of each phase are stabilized using a perspective transform based on sparse correspondences. Their respective absolute difference images ΔI_{before} and ΔI_{after} indicate an invalid result if $\max\left(\Delta I_{after}(u_i, v_i), \Delta I_{before}(u_i, v_i - d_i)\right) > \Delta I_{max}$.

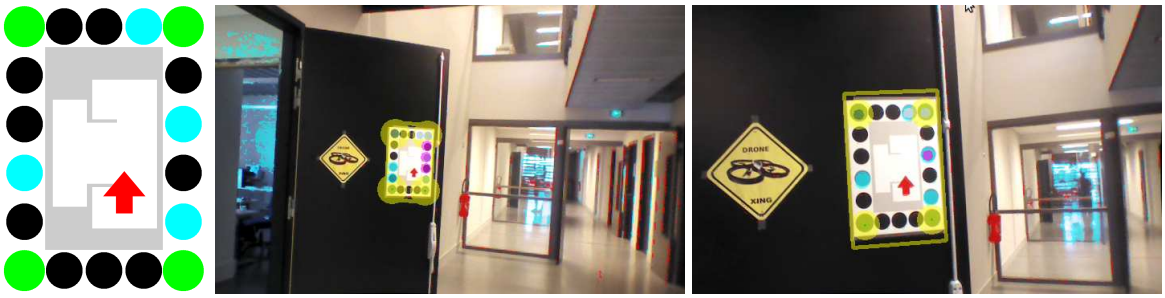

Figure 10: Landmark with machine-readable dot pattern and human-readable floor plan (left). Approach towards a detected yet un-recognized landmark (middle). Recognized landmark just before re-orientation for further flight (right).

6 Landmark-based Navigation

As explained before, our system is capable of navigating locally without any artificial landmarks, but requires them e. g. for taking previously defined turns at corridor crossings and junctions[2]. Fig. 10 shows an example of the currently employed wall-mounted markers, which have been developed in a student project [5]. They are detected and recognized via thresholding in the HSV color space: Detection only involves the green corners' relative positions, from which the landmark's distance and point of view can be derived as well. The 14 cyan/black dots along the edges allow recognizing $2^{12} = 4096$ individual markers, while two remaining bits are reserved for error detection via checksums.

We use this pattern to encode a location's ID and the quadcopter's inbound direction within each landmark. A supplementary mission description concisely maps all relevant crossings to outbound directions for further flight, and thereby yields a fully-defined path. Actual flight control is a two-stage process: Once a landmark is initially detected, it is rarely immediately recognized as well. However, the quadcopter usually has not yet entered the crossing or junction at this point either. In a 1^{st} stage, we therefore compute and approach a position directly facing the landmark at a distance of 1 m. The PID controllers of all 4 degrees of freedom are used cooperatively for this task. A 2^{nd} stage exclusively uses closed-loop yaw control to re-orientate the quadcopter towards its designated outbound direction. Sparse 3D reconstruction and "corkscrew" flight continue directly after it has been correctly aligned.

7 Conclusions and Outlook

We have presented each module of a complete system which has proven capable of autonomous indoor navigation. Despite the absence of any inherent 3D sensor on our quadcopter, it is able to perform metric 3D reconstructions mainly based on a monocular camera. Comparing the two complementary approaches we implemented for this purpose, the sparse method seems more suitable for the task of following way points: Its resulting 3D points consistently cover the area ahead and are continuously updated, while the dense method periodically needs to interrupt regular flight. Nevertheless, the latter's much larger number of result pixels may well be useful e. g. for mapping a building. For that application, visual odometry – merely an intermediate result at present – would also be of greater interest. Finally, the presented landmarks effectively provide topological localization and offer a convenient way to define the quadcopter's path.

[2]Without a landmark, the above strategies would heuristically follow the path requiring the least sharp turn.

Even though it has been sufficient for our application, the quality of sparse 3D reconstruction and visual odometry can be improved by applying bundle adjustment techniques to longer feature tracks. Their computational complexity might however inhibit compatibility with embedded on-board PCs. The localization and prediction of moving objects using a monocular camera requires resolving individual scale ambiguities and i. a. therefore still poses research challenges. Furthermore, visual place recognition such as [12] offers the opportunity to avoid the need for specific landmarks and to make our system fully independent from any infrastructure.

References

[1] 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*, pages 3056–3063, 2011.

[2] L. Apvrille. Autonomous navigation of micro drones. *youtube.com/watch?v=tamYpmGvzRw*, 2013.

[3] P. Bristeau, F. Callou, D. Vissière, and N. Petit. The navigation and control technology inside the ar.drone micro uav. In *IFAC World Congress*, pages 1477–1484, 2011.

[4] D. C. Brown. Decentering distortion of lenses. *Photometric Engineering*, 32(3):444–462, 1966.

[5] J. de Campredon. Report: Early image processing tests on uav (cont.). *Student Project at Institut Eurécom, Sophia Antipolis, France*, 2013.

[6] 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*, pages 2815–2821, 2012.

[7] 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*, pages 4557–4564, 2012.

[8] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium*, pages 963–968, 2011.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] G. Heppner, A. Roennau, and R. Dillman. Enhancing sensor capabilities of walking robots through cooperative exploration with aerial robots. *Journal of Automation, Mobile Robotics & Intelligent Systems*, 7(2), 2013.

[11] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[12] H. Lategahn, M. Schreiber, J. Ziegler, and C. Stiller. Urban localization with camera and inertial measurement unit. In *IEEE Intelligent Vehicles Symposium*, pages 719–724, 2013.

[13] A. S. McCormack and K. R. Godfrey. Rule-based autotuning based on frequency domain identification. *IEEE Trans. on Control Systems Technology*, 6(1):43–61, 1998.

[14] M. Monajjemi. Autonomylab/ardrone_autonomy. *github.com/AutonomyLab/ardrone_autonomy*, 2013.

[15] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *Journal of Intelligent and Robotic Systems*, 61(1-4):287–299, 2011.

[16] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *IEEE International Conference on Robotics and Automation, Workshop on Open Source Software*, 2009.

[17] Parrot SA. Ar.drone 2.0. parrot new wi-fi quadricopter- specifications. *ardrone2.parrot.com/ardrone-2/specifications*, 2013.

[18] C. Yuan, F. Recktenwald, and H. A. Mallot. Visual steering of uav in unknown environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3906–3911, 2009.