Robot Operating System (ROS), The Complete Reference (Volume 2), Springer 2017.

Autonomous Navigation in a Warehouse with a Cognitive Micro Aerial Vehicle

Marius Beul, Nicola Krombach, Matthias Nieuwenhuisen,
David Droeschel, and Sven Behnke

Autonomous Intelligent Systems Group, University of Bonn,
Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
mbeul@ais.uni-bonn.de, www.ais.uni-bonn.de

Abstract. Micro aerial vehicles (MAVs), such as multirotors, are envisioned for autonomous inventory-taking in large warehouses. Fully autonomous operation of MAVs in such complex 3D environments requires real-time state estimation, obstacle detection, mapping, and navigation planning. To this end, we employ a cognitive MAV equipped with multiple sensors including a dual 3D laser scanner, three stereo camera pairs, an IMU, an RFID reader, and a powerful onboard computer running the ROS middleware. Tasks with hard real-time requirements such as attitude control and state estimation are processed on a Pixhawk Autopilot, which communicates with the main computer via the MAVLink protocol. In this chapter, we describe our integrated system for autonomous MAV-based inventory in warehouses. We detail the involved components and evaluate our system with the real autonomous MAV in a realistic scenario. We also report lessons learned during field testing.

Keywords: MAV, Multimodal Sensor Setup, 3D Laser Scanner, Sensor Fusion, Autonomy

1 Introduction

Micro aerial vehicles (MAVs) are enjoying increasing popularity, both in research and in applications such as aerial photography, inspection, surveillance, and search and rescue missions. Most MAVs are remotely controlled by a human operator or follow global navigation satellite system (GNSS) waypoints in obstacle-free heights. For autonomous navigation in complex 3D environments, sufficient onboard sensors and computing power are needed in order to perceive and avoid obstacles, build 3D maps of the environment, and plan flight trajectories.

In this chapter, we present a use case for indoor MAV operation employing the ROS infrastructure: autonomous warehouse inventory. For this purpose, we built an MAV with a multimodal omnidirectional sensor setup, a fast onboard computer, and a robust data link. The sensors include a lightweight dual 3D laser scanner, three stereo cameras, and a radio-frequency identification (RFID) reader module. All components are lightweight and hence well suited for MAVs.

Fig. 1. Our MAV has been designed for inventory and short-range inspection tasks in indoor environments. Reliable perception of obstacles in the surrounding is key for safe operation.

Our MAV can localize itself in indoor environments fusing visual odometry and 3D laser scan registration to a 3D map. It avoids static and dynamic obstacles perceived with the onboard sensors reliably.

On the MAV, we employ ROS Indigo Igloo as middleware on top of Ubuntu 14.04 to facilitate fast development through a modular software design. This allows us to transfer technology between multiple MAVs, e.g., built for outdoor mapping [4], and ground robots, e.g., in a space exploration scenario [30]. Furthermore, ROS allows for an easy and flexible connection with several ground control stations.

After a discussion of related work in the next section, we will briefly describe our MAV. Our perception pipeline is explained in Sec. 4, putting special emphasis on the cameras (Sec. 4.2) and laser scanner (Sec. 4.3). We then outline our mapping approach in Sec. 5 and describe the localization and state estimation capabilities in Sec. 6. The navigation pipeline is detailed in Sec. 7. Sec. 8 describes the user interfaces. The MAV system is experimentally evaluated in Sec. 9. We conclude the chapter with a discussion of lessons learned in Sec. 10.

2 Related Work

In recent years, many MAVs with onboard environment sensing and navigation planning have been developed. Due to the limited payload of MAVs, most approaches to obstacle avoidance are camera-based [8, 17, 21, 24, 26, 28, 29, 33]. Approaches using monocular cameras to detect obstacles require translational movement in order to perceive the same surface points from different perspectives. In order to estimate depth of object points instantaneously, stereo cameras are used on MAVs, e.g., in the works of Schmid et al. [29] and Park and Kim [24]. Tripathi et al. [33] use stereo cameras for reactive collision avoidance. The limited field of view (FoV) of cameras poses a problem when flying in constrained spaces where close obstacles can surround the MAV.

To overcome these limitations, some MAVs are equipped with multiple (stereo) cameras. Schauwecker and Zell [28] use two stereo cameras, one oriented forward, the other backward. Moore et al. [20] use a ring of small cameras to achieve an omnidirectional view in the horizontal plane, but rely on optical flow for velocity control, centering, and heading stabilization only.

Grzonka et al. [11] use a 2D laser scanner to localize the MAV in environments with structures in flight altitude and to avoid obstacles. This limits obstacle avoidance to the measurement plane of the laser scanner. Other groups combine laser scanners and visual obstacle detection [13,14,32]. Still, their perceptual field is limited to the apex angle of the stereo camera (facing forward), and the mostly horizontal 2D measurement plane of the scanner. They do not perceive obstacles above or below this region or behind the vehicle. We allow omnidirectional 4D movements (3D position and yaw) of our MAV, thus we have to take obstacles in all directions into account. The proposed MAV extends our own previous work [4], an MAV with a 3D laser scanner and two wide-angle stereo camera pairs. Another MAV with a sensor setup that allows omnidirectional obstacle perception is described by Chambers et al. [3]. We significantly increase field of view and bandwidth of the onboard cameras, add a second laser scanner to measure simultaneously in orthogonal directions, and use a faster onboard computer.

In combination with accurate pose estimation, laser scanners are used to build 3D maps. Fossel et al. [9], for example, use Hector SLAM [15] for registering horizontal 2D laser scans and OctoMap [12] to build a three-dimensional occupancy model of the environment at the measured heights. Morris et al. [22] follow a similar approach and in addition use visual features to aid state estimation. Still, perceived information about environmental structures is constrained to lie on the 2D measurement planes of the moved scanner. In contrast, we use a continuously rotating laser scanner that does not only allow for capturing 3D measurements without moving, but also provides omnidirectional obstacle sensing at comparably high frame rates (4 Hz in our setup).

To the knowledge of the authors, there exist no scientific works regarding MAV-based stocktaking. However, it is worth mentioning that the proposed system was developed for the German BMWi funded Autonomics for Industry

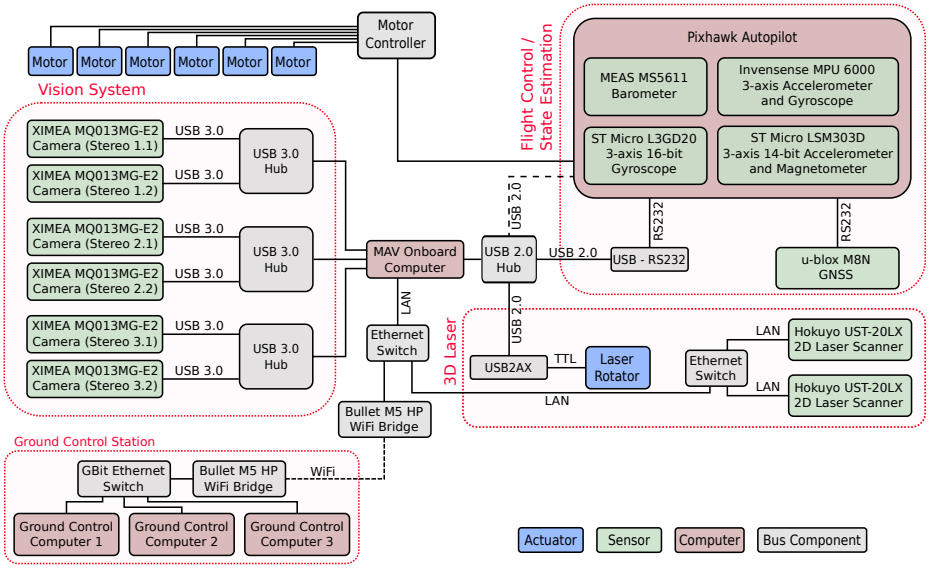


Fig. 2. Scheme of the sensors, actuators, computers, and bus systems on our MAV. We use high-bandwidth USB 3.0 connections for the cameras due to the high data rates, and lower-bandwidth buses for flight control and RFID reader. The dashed line indicates a wireless connection.

4.0 project InventAIRy [7]. Furthermore, the company DroneScan [25] recently demonstrated first results of MAV-based inventory.

3 System Overview

Our MAV design is a hexarotor with a 1.24 m diameter frame surrounding the rotor plane. The total weight is 5.0 kg. The thrust-to-weight ratio is approximately 1.5. Fig. 1 shows our MAV in an indoor environment. While fragile equipment like computer and laser scanner lies in the core of the MAV, the frame protects the rotors and is used for mounting multiple sensors. For sensor data processing and navigation planning, we use an unboxed Gigabyte GB-BXi7-4770R as the onboard processing system. The small board is equipped with an Intel Core i7-4770R quadcore CPU, 16 GB DDR3-memory, and a 480 GB SSD.

For state estimation, obstacle detection, localization, and mapping, our MAV is equipped with a multimodal sensor setup. Fig. 2 gives an overview of the installed sensors. The vision system of our MAV features a ring of six Ximea MQ013MG-E2 1.3 M Pixel USB 3.0 cameras, yielding an omnidirectional FoV. The cameras are used for visual odometry and for the detection of visual features like AprilTags [23].

We use two rotating Hokuyo UST-20LX laser scanners with orthogonal measurement planes to achieve a comprehensive perception of the MAV surround-

ings. Each laser scanner provides distance measurements of up to 20 m with 270° apex angle. The 3D laser is used for obstacle perception and 6D self-localization in a 3D map. An RFID reader module allows for the fast detection of passive RFID tags that identify storage places and warehouse stock.

For high-level navigation tasks, we employ ROS as middleware on the onboard computer and on ground control stations. For low-level velocity and attitude control, the MAV is equipped with a Pixhawk Autopilot flight control unit [19] that also contains gyroscopes, accelerometers, a compass, a barometer, and an optional GNSS receiver. We modified its firmware to meet our requirements. In contrast to the original implementation, we control the MAV by egocentric[1] velocity commands calculated by the onboard PC. Hence, we need a reliable egocentric velocity estimate, independent from allocentric[2] measurements, i.e., compass orientation. Our state estimation filter, which estimates 3D positions, 3D velocities, and 3D accelerations, integrates—in addition to the measurements already considered in the original implementation—external sources provided by the onboard PC. These include visual odometry velocities and laser-based localization.

To achieve high camera frame rates at full resolution, we connect the cameras via three hubs, one per stereo pair, to a dedicated USB 3.0 bus of the onboard computer. Onboard components with lower bandwidth requirements, i.e. the flight control unit and the laser scanner rotator, are connected to a second USB 2.0 bus. The Pixhawk Autopilot is connected twice. The first connection via an USB-to-serial converter provides the telemetry and control connection according to the MAVLink protocol [18]. The second connection is inactive during flight and is only used for debugging and firmware updates of the Pixhawk Autopilot on the ground. Fig. 2 illustrates our onboard USB setup.

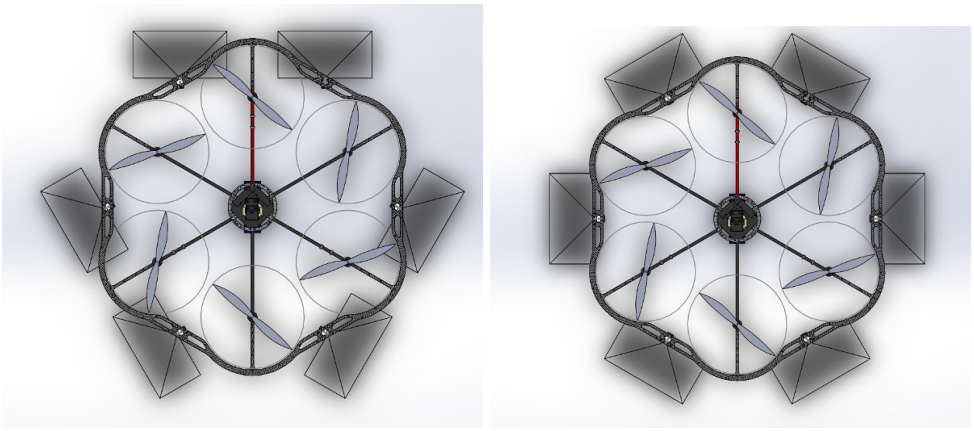
While our MAV frame also supports the use of 15" propellers, we use six MK3644/24 motors (111 g each) with 14" propellers to generate thrust. Turnigy 5S, 10 Ah, 35C batteries power the MAV, including all periphery. The batteries weight 1.28 kg each and are hot-swappable. Thus, it is not necessary to shut down the onboard computer while changing batteries.

Real-time debugging and control of onboard functions with our ground control stations is crucial for the efficient development of algorithms. To ensure seamless operation, we use two Ubiquity Networks Bullet BM5HP WiFi adapters. They are configured to work in Wireless Distribution System (WDS) Transparent Bridge Mode to behave as if a wired connection would be present. Our network setup is also shown in Fig. 2.

4 Perception

We use the ROS packages `tf`, `robot_state_publisher`, and `urdf` to incorporate the physical sensors, mounted on the MAV, into our software. The transformations for the robot model are first estimated from a coarse CAD model (Fig. 3,

[1] The egocentric frame lies in the center of the MAV.
[2] The allocentric "world" frame is a globally fixed frame in the warehouse.



(a) Triple stereo configuration (b) Omnidirectional configuration

Fig. 3. Mounting of the cameras in (a) triple stereo and (b) omnidirectional configuration. Configuration (a) facilitates the usage of available standard stereo methods. In configuration (b) cameras have partial image overlap with both neighboring cameras. This allows the development of truly omnidirectional vision methods.

Fig. 6), and later calibrated by sensor-specific methods which are described in the respective subsections. In the following, we detail the sensors used on the MAV, and describe how they are incorporated into our ROS infrastructure.

4.1 Accelerometers, Gyros, Compass, and Barometer

Low-level sensors like accelerometers, gyros, compass, and barometer are part of the Pixhawk Autopilot to ensure real-time processing of these—in relation to, i.e., USB interface latency—comparatively fast sensors. For a fast transient response, state estimation—detailed in Sec. 6—runs directly on the Pixhawk Autopilot. Hence, raw data of accelerometers, gyros, compass and barometer is only fed to the main computer for logging purposes (e.g., `sensor_msgs/Imu` ROS message), but processed directly on the flight control unit. The filtered results are also transferred to the onboard PC with the MAVLink protocol and published on ROS topics by our ROS-MAVLink communication node which is based on `mavlink_ros`[3].

4.2 Cameras

We use six Ximea MQ013MG-E2 global-shutter monochrome USB 3.0 cameras (22.5 g each) for visual perception. The camera configuration can be easily switched from three stereo-pairs (Fig. 3a) to an omnidirectional configuration including all six cameras (Fig. 3b). While the stereo configuration facilitates the

[3] `http://github.com/mavlink/mavlink_ros`

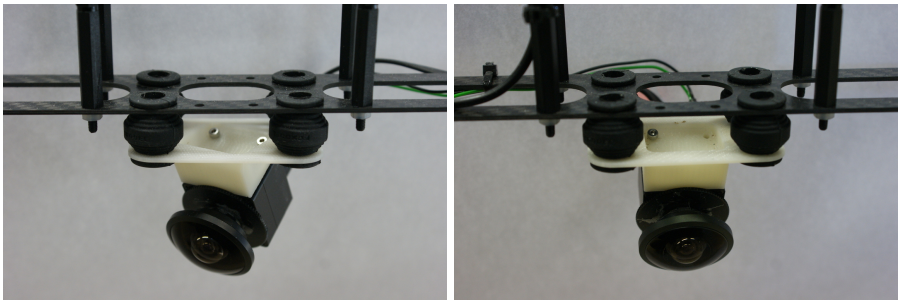


Fig. 4. To cope with unavoidable vibrations during flights, our camera mounts are equipped with vibration dampers. The left mount is for stereo configuration, the right mount for omnidirectional camera configuration.

usage of available standard stereo methods, the latter allows the deployment of truly omnidirectional vision methods that have not been addressed by many researchers yet. The mountings are detailed in Fig. 4. We use vibration dampers to isolate the cameras from high frequency oscillations caused by the imbalance of the propellers. Each mounting—including dampers—weights 11.5 g.

In combination with Lensagon BF2M2020S23 fisheye lenses with 195° apex angle (25 g each), an omnidirectional FoV can be obtained. The use of multiple camera pairs not only facilitates omnidirectional obstacle perception, but also provides redundancy. So if, e.g., the MAV points one camera-pair towards featureless surfaces or the sky, the others are still able to perceive the environment.

Communication and control With every pair of stereo cameras sharing a passive USB 3.0 HUB, we achieve frame rates of up to 55 fps, depending on exposure timings. All cameras are synchronized by hardware triggering. When data from all cameras has been received, the next frame is triggered. This enables us to achieve adaptive high frame rates which results in data rates of up to 200 MB/s. Fig. 5 shows an image obtained during flight.

The communication with all six cameras is performed by a single driver node that processes all images. The node acts as a wrapper for the Ximea `xiAPI`[4] API. Missing on-camera functionalities, e.g., gamma correction and rectification, are performed directly in this node and the enhanced fisheye and rectified images are published on ROS topics, accordingly. We use the `dynamic_reconfigure` ROS package to set the camera and acquisition parameters, e.g. exposure time, gain, frequency, and auto-exposure, dynamically during runtime. Additionally, the user has the choice to employ different image enhancement techniques, as listed in Tab. 1. The image correction and rectification is performed on the original 10-bit images during readout, using a pre-generated look-up table. By using the `dynamic_reconfigure` server, the look-up table can be recalculated if the correction parameters are changed by the user. The processed images are published

[4] `https://www.ximea.com/support/wiki/apis/XiAPI`

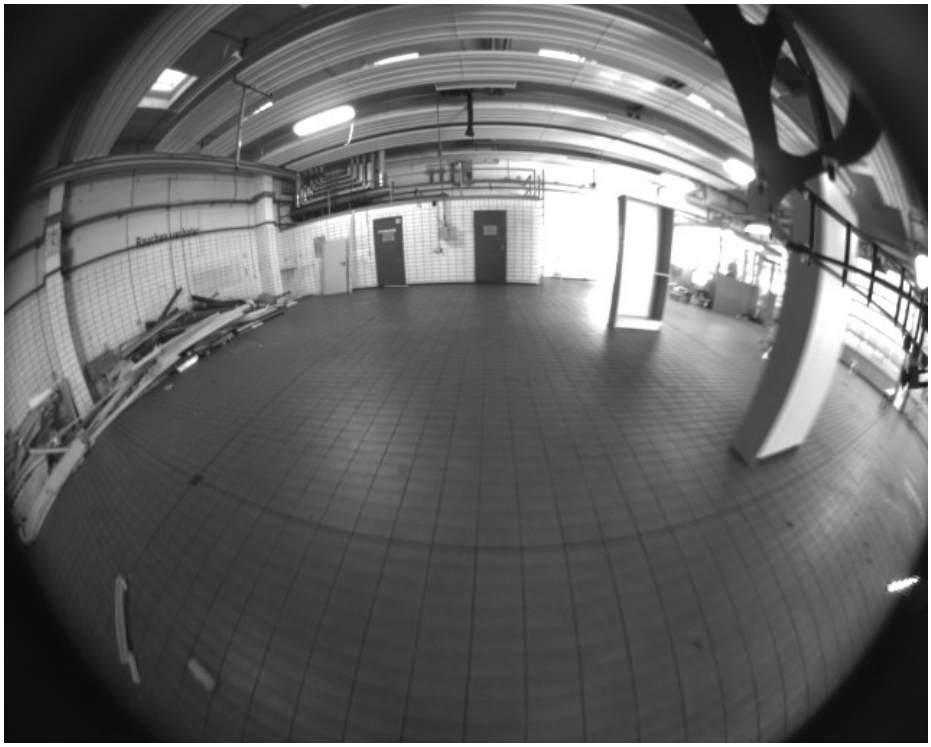


Fig. 5. Fisheye camera image of one of the onboard cameras. Due to the large FoV, the frame is slightly visible in the stereo configuration. We address this issue by either masking or by rectification which removes these artifacts.

as `sensor_msgs/Image` messages, together with downsampled rectified images and corresponding `sensor_msgs/CameraInfo` info messages. For efficiency reasons, the driver can write camera Bag files directly to the disk, bypassing the ROS communication infrastructure.

Calibration and rectification Each stereo camera pair is calibrated intrinsically and extrinsically, using the epipolar equidistant model [1] especially developed for fisheye camera calibration. It uses a 3D calibration target with point markers on three orthogonal planes (see Fig. 6b), which is observed from different distances and angles during calibration. The approximate positions of the 3D point markers have to be known. In an offline calibration run, the intrinsic and extrinsic calibration parameters are estimated together with the 3D coordinates of the calibration target by bundle adjustment, formulated as least squares problem. The user can select different applicable projection and distortion models in the calibration toolbox. For modeling the projection of our fisheye cameras, we employ the epipolar equidistant model, that describes the projection

Table 1. Available pixel-wise operations for image I.

Image transformation	Operation
Gamma	$I_{out} = c \cdot I_{in}^{\frac{1}{\gamma}}$
Logarithmic	$I_{out} = c \cdot \log(1 + I_{in})$
Contrast-stretching	$I_{out} = \frac{1}{1+\left(\frac{m}{I_{in}}\right)^E}$
10-to-8-bit conversion	$I_{out} = I_{in} \gg 2$
None	$I_{out} = I_{in}$

The parameters, including gamma γ, scaling constant c, mid-line m, and slope control E, can be changed during runtime using, e.g., `rqt_reconfigure`.

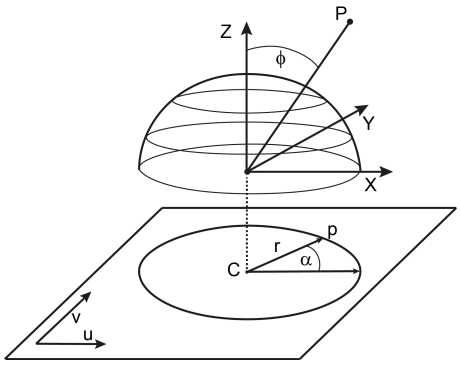
Table 2. Rectification runtime.

Target image resolution	Time per image
1280 × 1024	4 ms
640 × 512	1 ms
320 × 256	0.7 ms

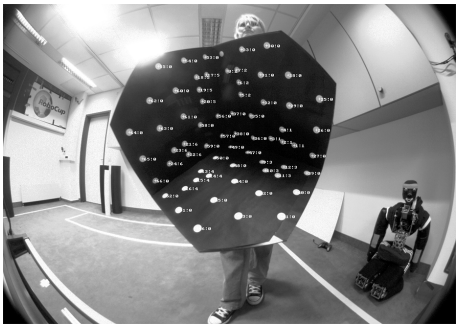
of a spherical image onto a plane as shown in Fig. 6a. The lens distortion is described using a third-order Chebyshev polynomial. For image rectification with horizontal epipolar lines, we pregenerate look-up tables to allow for fast online processing during flight. Furthermore, to improve computational efficiency, the images are downsampled to half the resolution during rectification. The overall time needed for the rectification of one image is approximately 1 ms. Timings for different resolutions are listed in Tab. 2. Overall, we obtain a reprojection error of 0.75 pixels and estimate a baseline of 53.362 cm.

4.3 Laser Scanner

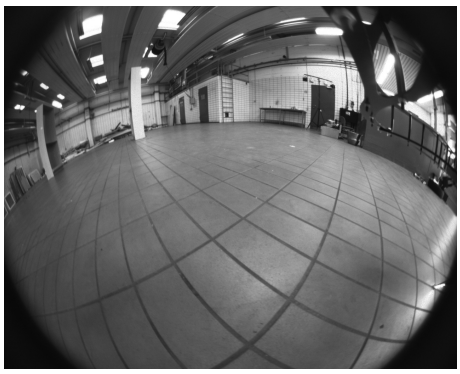
Our custom-built 3D laser perceives the environment around the MAV at a frequency of 2 Hz. The sensor combines two Hokuyo UST-20LX laser range finders mounted on a link. A Robotis Dynamixel MX-28 servo actuator rotates the link around the vertical axis with one revolution per second, yielding an spherical FoV. The servo actuator measures the angular position of the laser range finders with 0.088° resolution. Fig. 6 depicts the scanner arrangement, showing that one scanning plane is parallel to the axis of rotation while the other is twisted by 45° to obtain denser measurements in a ±45° vertical × 360° horizontal FoV. This arrangement results in a small upward pointing cylindrical blind spot of the first scanner and conical, upward and downward pointing blinds spot for the twisted scanner. Hence, this setup maximizes the FoV and obtains many measurements in flight height. Since the blind spot is closed by copter attitude changes, it does



(a) Fisheye camera model



(b) 3D point target



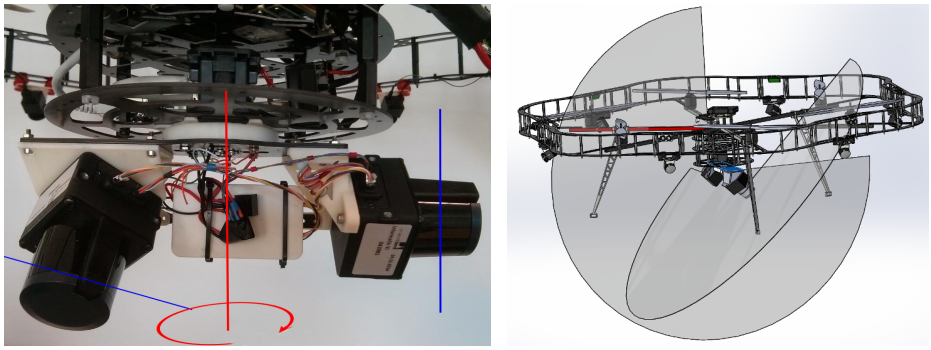
(c) Raw image

(d) Rectified image

not degrade mapping or obstacle detection in our scenario. Furthermore, due to the large FoV of 270° within the scan planes, a half rotation of the link produces a 3D scan in almost all directions.

Each 2D laser range finder has a scanning frequency of 40 Hz with 1,080 measurements per scan plane resulting in 43,200 measurements per second. Fig. 7 shows resulting point clouds of the environment perceived by each laser and the combined point cloud. Each scanner weights 143 g (without cables). The whole sensor assembly weights 420 g including motor, a network switch, and a slip ring allowing for continuous rotation. For communication with the two individual laser scanners, we employ the driver provided by the ROS `urg_node` package.

The wide FoV of the laser scanner inherently leads to many measurements on the MAV itself. Considering the complex structure of the MAV, with moving parts like propellers, we remove measurements that belong to the robot's body. This so-called *self filter* approximates the model of the MAV by a cylinder with the diameter and height of the MAV. Furthermore, we use a modified shadow filter—based on the ROS `laser_filters` package—to remove not only incorrect measurements at the edges of the geometry, but also erroneous measurements caused by the fast rotating propellers. Filtering results are shown in Fig. 8.



(a) Laserscanners Photo (b) Laserscanners CAD

Fig. 6. Photo and CAD drawing of our 3D laser scanner with the FoV of the individual 2D laser scanners (blue). (a) Scanner 1 (right), and scanner 2 (left) have different FoVs. The Hokuyo 2D laser scanners are mounted on a bearing and rotated around the red axis. (b) Scanner 1 is rotated to the back of the image plane to show the 270° opening angle of the scanner. Scanner 2 is in the front, showing the twisted scan plane.

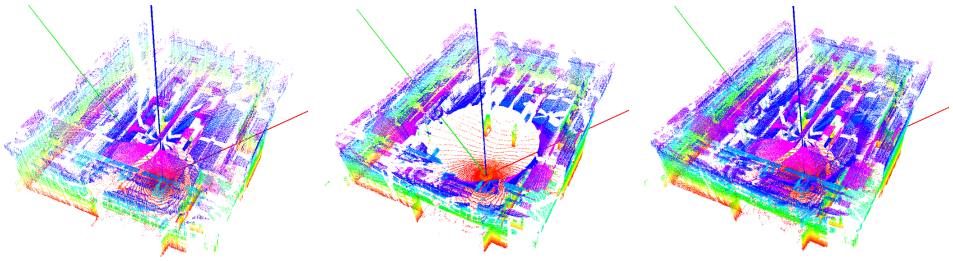
The absolute position of the laser relative to `base_link` is calibrated manually. The length of the link, the 2D laser scanners are mounted on, is crucial for scan consistency. Since it is only approximately known, we iteratively tune this parameter. By visualizing all single scanlines of a whole 3D scan in RViz, the parameter can be adjusted, until walls and ceilings have low variance.

We construct an MAV-centric multiresolution grid map that is used to accumulate sensor measurements [5]. We first register newly acquired 3D scans with the so far accumulated map and then update the map with the registered 3D scan. The map is utilized by our path planning and obstacle avoidance algorithms described in subsequent sections.

3D Scan Assembly When assembling 3D scans from raw laser scans, we account for the rotation of the scanner w.r.t. the MAV and for the motion of the MAV during acquisition. Thus, scan assembling mainly consists of two steps.

First, measurements of individual scan lines are undistorted with regards to the rotation of the 2D laser scanner around the servo rotation axis (red axis in Fig. 6). Here, the rotation between the acquisition of two scan lines is distributed over the measurements by using spherical linear interpolation provided by `laser_geometry/LaserProjection`.

Second, we compensate for the motion of the MAV during acquisition of a full 3D scan. To this end, we incorporate a motion estimate from the low-level filters running on the Pixhawk incorporating inertial measurement unit (IMU) and visual odometry measurements. The 6D motion estimate is used to assemble the individual 2D scan lines of each half rotation to a 3D scan.



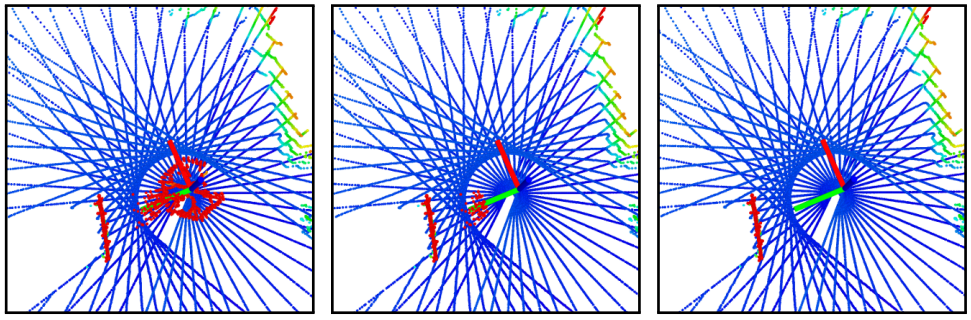
(a) Laser scanner 1 (b) Laser scanner 2 (c) Combined scans

Fig. 7. Point clouds from the rotating 3D laser scanner. While the individual scanners show substantial blind spots, nearly no occlusions occur in the combined scan. (b) Especially laser scanner 2 shows a large blind spot above the MAV caused by the limited opening angle and the twisted mounting position. See also Fig. 6. The axes represent the pose of the MAV. Color encodes height.

Local Multiresolution Map The assembled 3D scans are aggregated in a local multiresolution grid map [5]. Local multiresolution maps have a high resolution close to the robot and a lower resolution farther away. Each grid cell represents both occupancy information and the most recent individual distance measurements. The measurements of each cell are summarized in a surface element (surfel) by the sample mean covariance (cf. Fig. 9). Compared to uniform grid-based maps, multiresolution leads to the use of fewer grid cells—without losing relevant information—and consequently results in lower computational costs. Fig. 9 shows an example of our local multiresolution grid-based map.

Registration Approach We register each newly acquired 3D scan with the local multiresolution map of the environment with our surfel-based registration method [5]. Instead of considering each point individually, we represent the 3D scan as local multiresolution grid and match surfels. A newly acquired scan (scene) is aligned to the local multiresolution map (model) by finding a rigid 6 degree-of-freedom (DoF) transformation $T(\theta)$ that best aligns the scene surfels to the model surfels.

Compared to dense RGB-D images [31] or high-resolution static 3D laser scans used in our previous work [27], 3D scans obtained from our laser scanner are much sparser. We cope with this sparsity through probabilistic assignments of surfels during the registration process. Observations are described by a mixture model, avoiding hard associations between surfels. The transformation $T(\theta)$ is recovered by *expectation maximization* (EM), where the E-Step finds new surfel assignments based on the last estimation of θ and the M-step optimizes θ based on the last assignments. This optimization is efficiently performed using the Levenberg-Marquardt (LM) method as in [31]. By summarizing measurements in surfels, and therefore considering significantly less elements for registration, we



(a) Unfiltered (b) Self filter (c) Self and shadow filter

Fig. 8. Demonstration of the employed scan filters. A 3D scan assembled from one half rotation of the 3D laser scanner is shown from a top-view. Color encodes height. The MAV (depicted by the axes) passes the obstacle on the left. The red points close to the MAV are spurious measurements caused by the MAV itself and the occluded transition between the obstacle and the MAV. (a) Unfiltered 3D scan. (b) Filtered 3D scan using the self filter only. Spurious measurements remain. (c) Filtered 3D scan using self filter and modified shadowing filter. Spurious measurements are removed.

gain efficiency. When matching surfels, we choose the finest common resolution available between both maps to achieve accuracy.

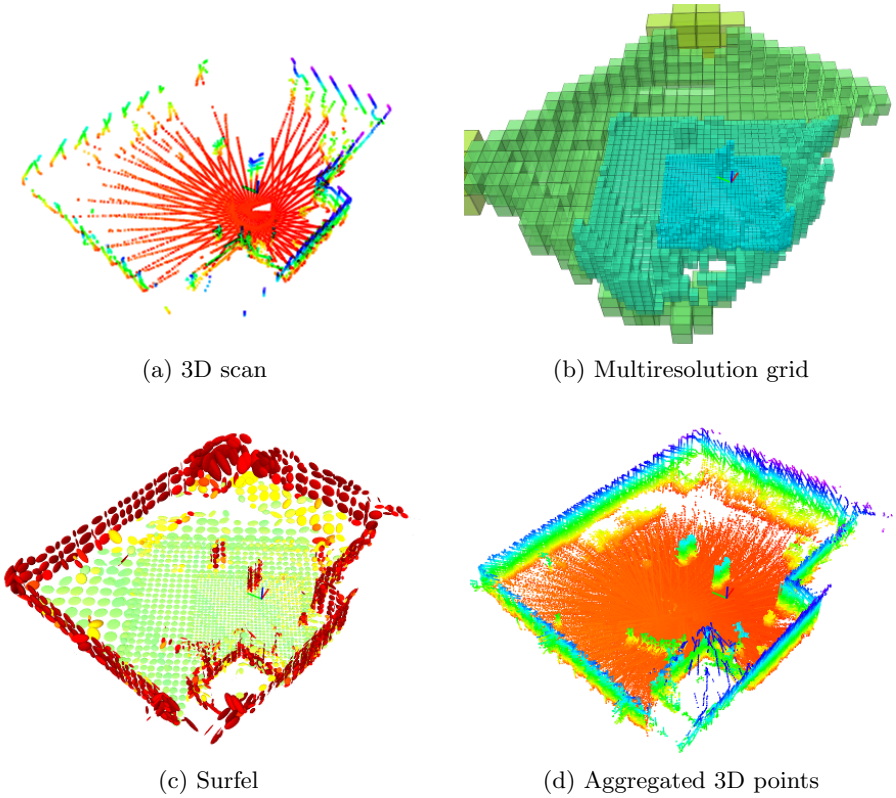
4.4 Radio-frequency Identification

We inventory stock either by visually perceiving and mapping attached April Tags (Sec. 6.3) or by locating attached RFID Tags. To read RFID tags placed on shelves or inventory, our MAV is equipped with a ThingMagic M6e RFID module and an unboxed SkyeTek SP-AN-04-UF-BB6LP directional antenna (Fig. 10). The module can detect RFID tags at distances up to several meters, depending on transmit power. A ROS node, based on the ThingMagic `Mercury API`[5], decodes the RFID readings and converts them to ROS messages containing a header, the detected ID as string, and a signal strength indicator. Together with the MAV pose, these messages could be sent to a warehouse management system (WMS). To this end, we project received RFID detections into the allocentric warehouse map by means of a simple sensor model for visualization purposes.

5 Mapping

For fast estimation of the MAV motion, we incorporate IMU and visual odometry measurements into velocity and pose estimates. While these estimates allow us to control the MAV and to track its pose over a short period of time, they are prone to drift and thus are not suitable for localization on the time scale

[5] `http://www.thingmagic.com/index.php/mercuryapi`



(a) 3D scan (b) Multiresolution grid

(c) Surfel (d) Aggregated 3D points

Fig. 9. Local multiresolution grid map. (a) The 3D scan acquired with our continuously rotating laser scanner, ceiling removed for better visibility. (b) The multiresolution grid structure of the map. Cell size (indicated by color) increases with the distance from the robot. (c) For every grid cell a surfel summarizes the 3D points in the cell. Color encodes the orientation of the surfel. (d) 3D points stored in the local multiresolution map. Color encodes height from ground.

of a mission. Furthermore, they do not provide a fixed allocentric frame for the definition of mission-relevant poses independent from the MAV. Thus, we build an allocentric map by means of laser-based simultaneous localization and mapping (SLAM) before mission execution and employ laser-based pose tracking w.r.t. this map during autonomous operation.

This allocentric map is built by aligning multiple local multiresolution maps, acquired from different view poses [6]. We model the different view poses as nodes in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that are connected by edges. A node consists of the local multiresolution map from the corresponding view pose. Each edge in the graph models a spatial constraint between two nodes.

After adding a new 3D scan to the local multiresolution map as described in Sec. 4.3, the local map is registered towards the previous node in the graph

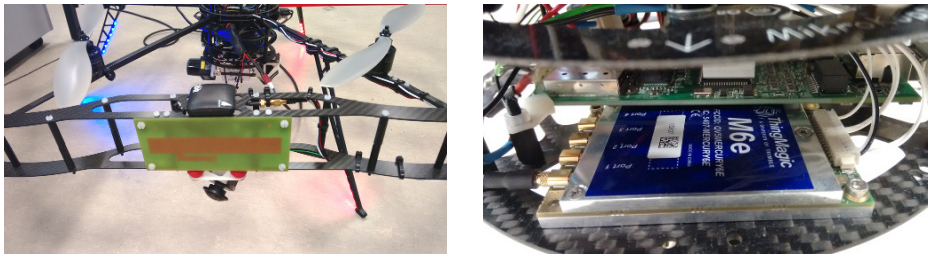


Fig. 10. RFID sensor. The MAV is equipped with a lightweight RFID antenna (left) and a small RFID reader module (right), connected to the onboard PC via USB. The RFID system is used to map positions of RFID tags in the allocentric map attached to shelves or inventory.

using the multiresolution surfel registration with probabilistic assignments [5]. A new node is generated for the current local map, if the MAV moved sufficiently far. The registration result x_i^j between a new node v_i and the previous node v_j is a spatial constraint that we maintain as values of edges $e_{ij} \in \mathcal{E}$. In addition to edges between the previous node and the current node, we add spatial constraints between close-by view poses that are not in temporal sequence.

On each scan update, we check for one new constraint between the current reference v_{ref} and other nodes v_{cmp}. We determine a probability

$$p_{\text{chk}}(v_{\text{cmp}}) = \mathcal{N}\left(d(x_{\text{ref}}, x_{\text{cmp}}); 0, \sigma_d^2\right)$$

that depends on the linear distance $d(x_{\text{ref}}, x_{\text{cmp}})$ between the view poses x_{ref} and x_{cmp}. According to $p_{\text{chk}}(v)$, we choose a node v from the graph and determine a spatial constraint between the nodes using our surfel registration method.

From the graph of spatial constraints, we infer the probability of the trajectory estimate given all relative pose observations

$$p(\mathcal{V} \mid \mathcal{E}) \propto \prod_{e_{ij} \in \mathcal{E}} p(x_i^j \mid x_i, x_j).$$

Each spatial constraint is a normally distributed estimate with mean and covariance determined by our probabilistic registration method. This pose graph optimization is efficiently solved using the `libg2o` ROS package by Kuemmerle et al. [16], yielding maximum likelihood estimates of the view poses x_i.

After the MAV has traversed the environment, the allocentric map is built from the optimized pose graph by merging all local surfel maps. Here, we use surfels with uniform resolution. Fig. 11 shows an example map acquired from a flight through a warehouse aisle. Our mapping pipeline is available as open-source ROS-based package[6].

[6] `https://github.com/AIS-Bonn/mrs_laser_map`

Fig. 11. SLAM point cloud. Left: Resulting point cloud after pose graph optimization acquired by a manual flight along a warehouse aisle (color depicts height). Right: Photo of the mapped aisle.

6 Localization and State Estimation

In order to navigate in indoor and outdoor environments, robust localization and state estimation, especially in GNSS-denied environments, is crucial. Our multimodal localization and state estimation pipeline exploits the specific characteristics of all sensors in terms of, e.g., accuracy and speed.

6.1 Triple Stereo Visual Odometry

Our visual odometry estimation is based on the ROS `viso2` package that wraps the visual odometry library LIBVISO2 [10], a fast feature-based method for monocular and stereo cameras. The approach does not require a motion model. The only prerequisites are that the input images are rectified and the extrinsic camera calibration is known.

We rectify the fisheye images with the method *epipolar image rectification* on a plane with an equidistant model as proposed by Abraham and Förstner [1]. The resolution of the rectified images is 640×512. The rectified image pairs are fed into three instances of `viso2` running in parallel—one for each stereo camera pair—to obtain three velocity estimates.

Similar to other feature-based methods, `viso2` extracts and matches features over subsequent stereo frames and estimates the camera motion by minimizing the reprojection error. Four types of features (corners and blobs of two polarities) are detected using 5×5 filters and non-maximum suppression. Feature similarity is computed by sparse horizontal and vertical Sobel filters. As shown in Fig. 12, feature associations are searched in small prediction windows between frames and along epipolar lines between the stereo pairs and matches are only accepted if a circular match across two adjacent frames and the two cameras can be established. Based on all found circle matches, Geiger et al. [10] estimate the camera motion by minimizing the reprojection error using Gauss-Newton optimization in combination with RANSAC for outlier removal.

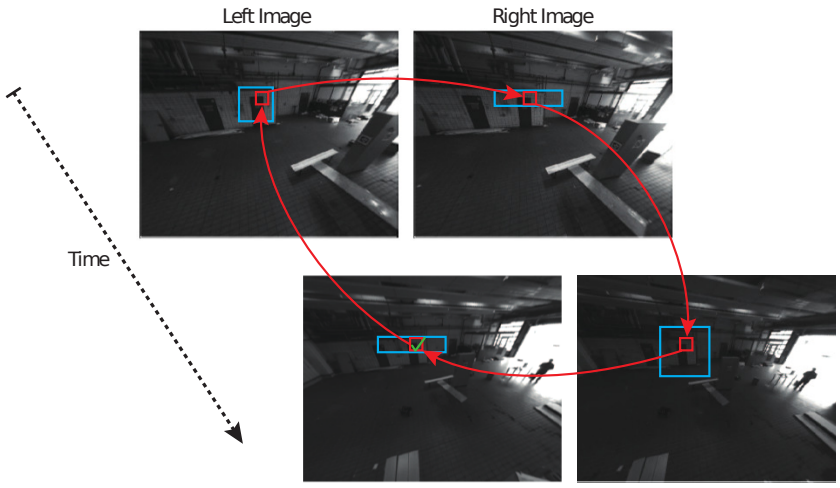


Fig. 12. Circular matching of feature points by `viso2` [10]: starting from a feature detected in the current left image (lower left), a windowed correspondence search (blue box) is performed on the previous left image (upper left). If a match has been found, it is matched along the epipolar line to the previous right image and from there to the current right image. The best match for this feature is searched along the epipolar line in the current left image. The match is accepted only if the loop is closed.

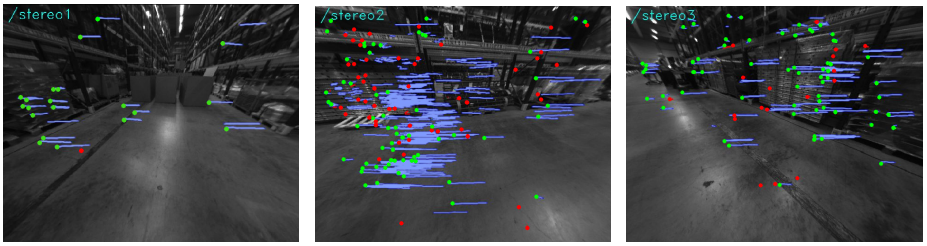
The estimated 3D velocities from the three stereo pairs are utilized in the state estimation pipeline. We weight each velocity estimate according to the number of correspondences that are tracked. When the number of features falls below a threshold, e.g. due to featureless or overexposed scenes, the weight is set to zero. In this way, we obtain visual odometry even if two cameras fail at the same time. Moreover, especially at fast forward motions where the feature correspondence search with the frontal camera is challenging, the estimates of the lateral cameras allow for proper motion estimation, as shown in Fig. 13.

The independent odometry estimates are published in the `base_link` coordinate frame. As the transformation from the camera coordinate systems to the `base_link` is static, it is looked up once at the beginning by using the TransformListener of the ROS `tf` package.

6.2 Laser-based Pose Tracking

In order to localize the robot in GNSS-denied environments, e.g., indoor environments, in an allocentric frame, we register local multiresolution maps to a global map employing multiresolution surfel registration (MRSR) [5]. In small environments, suitable maps can be built from the takeoff position before a mission. In larger environments, we perform laser-based SLAM (cf. Sec. 5).

Since the laser scanner acquires 3D scans with a relatively low rate of 2 Hz, we incorporate the egomotion estimate from the visual odometry and measurements



(a) Front stereo pair (b) Back-left stereo pair (c) Back-right stereo pair

Fig. 13. Triple stereo visual odometry. While the forward facing camera tracks few features due to fast forward motion, the remaining stereo pairs can still estimate reliable feature correspondences. Correspondences within one stereo pair are colored blue. Feature correspondences tracked by `viso2`. RANSAC is used for outlier detection. Inliers are colored green, outliers are colored red.

from the IMU to track the pose of the MAV. The egomotion estimate is used as a prior for the motion between two consecutive 3D scans. In detail, we track the pose hypothesis by alternating the prediction of the MAV movement given the filter result and alignment of the current local multiresolution map towards the allocentric map of the environment.

To align the current local map with the allocentric map, we also use the surfel-based registration described in Sec. 4.3. The allocentric localization is triggered after a new 3D scan has been registered with and added to the local multiresolution map. We update the allocentric robot pose with the resulting registration transform. To achieve real-time performance of the localization module, we only track one pose hypothesis. We assume that the initial pose of the MAV is known, either by starting from a predefined pose, or by means of manually setting the pose. Fig. 14 shows the registration of a 3D scan to the map and an estimated 6D trajectory.

The resulting robot pose estimate is used as a measurement update in a lower-level state estimation filter. We propagate this allocentric pose over time with visual odometry and IMU to obtain allocentrically consistent pose and velocity estimates at a sufficiently high rate for planning and control.

6.3 AprilTag Detection

In order to improve the indoor localization of our MAV in environments with repetitive structures, e.g., warehouses, and to localize tagged objects, we augment the environment with AprilTags. These tags can be robustly detected in real time with the wide-angle cameras. Fig. 15 shows the detection of AprilTags with 164 mm edge length. The algorithm is able to detect and locate tags in distances of 0.5 m to 5.0 m. The computation time is 10 ms per image. We build maps of AprilTags in an allocentric frame by mapping with known poses based

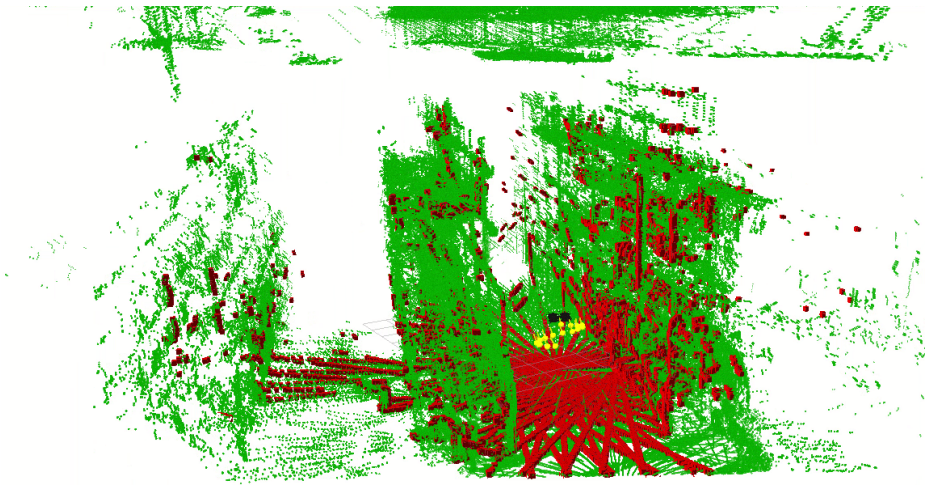


Fig. 14. Laser-based localization. A laser scan aggregated over 500 ms (red) is matched to an allocentric map (green) to track the MAV pose (black). The yellow dots depict the tracked MAV trajectory.

on laser-based localization. Fig. 15 also shows the resulting map after an example flight based on the observations from all six cameras.

6.4 State Estimation Filter

We use two filters for state estimation: A low-level extended Kalman filter (EKF) fuses measurements from accelerometers, gyros, and compass to one 6D attitude and acceleration estimate. The second, higher-level, filter fuses linear acceleration, velocity, and position information to a state estimate that includes 3D position. The low-level filter is supplied with the Pixhawk Autopilot. The higher-level filter extends the original Pixhawk Autopilot position estimator by incorporating all the sensors present on the MAV into one state.

Here, we predict the state:

$$x = \begin{pmatrix} p_x & p_y & p_z \\ v_x & v_y & v_z \\ a_x & a_y & a_z \end{pmatrix},$$

consisting of 3D position p, 3D velocity v, and 3D acceleration a under the assumption of uniform acceleration

$$\begin{aligned} p_k &= p_{k-1} + v_{k-1} \cdot dt + \frac{1}{2} a_k \cdot dt^2, \\ v_k &= v_{k-1} + a_k \cdot dt, \\ a_k &= a_{k-1}. \end{aligned}$$

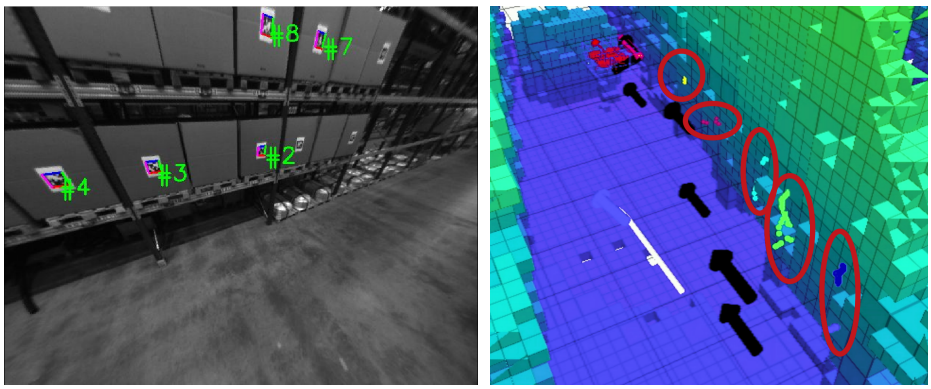


Fig. 15. AprilTag detections. Left: Detected AprilTags in the rectified camera image with corresponding ID. Right: Poses of the tag detections projected into the allocentric map with the MAV pose estimate before filtering. The colors correspond to the tag IDs. Black arrows depict mission view poses.

Table 3. Information sources for the state filter.

Information			Update	Frame	Weighting
Source	Type	Dim.	Rate (Hz)		Factor
Attitude EKF	Lin. Acceleration	3D	250	egocentric	20
Visual Odometry	Velocity	3D	15	egocentric	0-2
GNSS	Velocity	3D	10	allocentric	2
Barometer	Position	1D	250	allocentric	0.5
Laser Pose Tracking	Position	3D	2	allocentric	2
GNSS	Position	3D	10	allocentric	1

If sensor measurements are available, the state is corrected accordingly. For 1D velocity estimates $v_{k,sens}$, coming from, e.g., visual odometry, the state correction is

$$v_k = v_{k-1} + (v_{k,sens} - v_{k-1}) \cdot w \cdot dt,$$
$$a_k = a_{k-1} + (v_{k,sens} - v_{k-1}) \cdot w^2 \cdot dt^2.$$

Here, w is a weighting factor that indicates the reliability of the inputs. Table 3 shows the measurements that contribute to the filter result. Egocentric measurements are first transformed into the allocentric frame by the attitude estimate. We determined the weighting factors by iterative tuning.
This predictor/corrector design offers the following advantages. It

- delivers fast transient responses,
- works in GNSS-denied environments, and

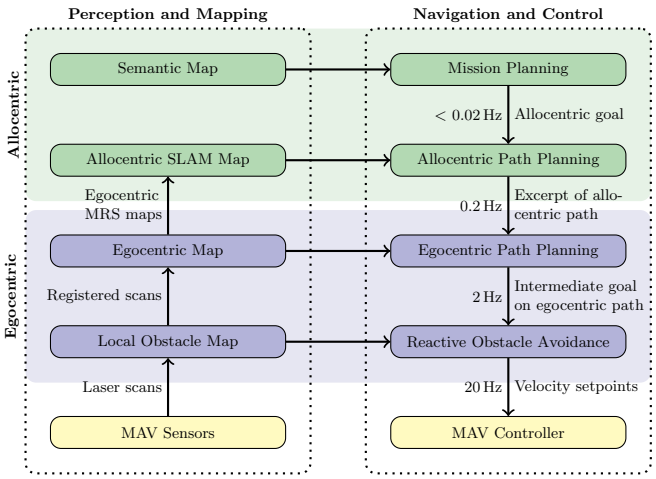


Fig. 16. Our navigation pipeline consist of five hierarchy levels. From top to bottom, the planning frequency increases, whereas the level of abstraction decreases.

– does not accumulate drift.

As can be seen in Fig. 2, we use a USB-to-serial converter to communicate with the Pixhawk Autopilot. We use the maximum rate of 921,600 baud to achieve a measurement frequency of up to 250 Hz for attitude, velocity, and position updates.

7 Navigation

To facilitate efficient and safe operations without or with only small human interaction, we employ the multilayered navigation approach illustrated in Fig. 16. Each layer operates in a frequency suitable for the specific task and on a correspondingly updated and accurate environment representation. From top to bottom these layers are: Mission planning, allocentric path planning, egocentric path planning, reactive collision avoidance, and low-level control. The planning frequency increases from top to bottom, whereas the level of abstraction decreases.

7.1 Mission Planning

The layout of large warehouses follows often a very structured pattern. Large shop floors are filled with shelves, containing standardized storage units, e.g., capable to store exactly one EUR-pallet of size 80×120 cm. Thus, on the topmost layer, we describe equal parts of a warehouse by number of shelves, unit height, and the numbers of units in horizontal and vertical direction. If the storage unit IDs are assigned in a systematic way, we can derive a mapping between storage unit coordinates, scan positions, and IDs automatically. Fig. 17 depicts such a

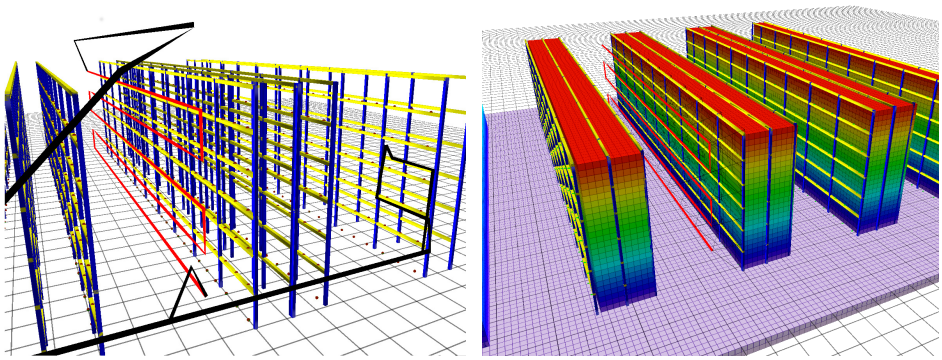


Fig. 17. Mission planning in semantic map. Based on warehouse parameters like shelf and storage unit dimensions, aisle width, etc., a semantic map of the shelves is generated. Dark red dots depict storage units. Left: An operator can command coverage tours to scan complete shelves (red path in left aisle), flights to specific storage locations (black path to right aisle), or a combination as part of a more complex flight plan. Right: To aid initial mission and path planning, we derive an OctoMap from the semantic map (color encodes height).

model. We derive an initial OctoMap from the model for navigation planning. For development and debugging, flight plans containing flights to individual storage units and coverage paths for whole shelves can be assembled using an RViz-based interface. In real-world applications, IDs of shelves or units to inspect will be provided by a warehouse management system (WMS).

Coverage paths to inventory shelves are generated from a user-defined distance to the shelf and the sensor apex angles. A 10 % overlap between scans allows detecting visual tags that could be cropped otherwise and mitigates the effects of small deviations from the flight altitude.

For missions involving flights to multiple individual storage units, we formulate the mission as traveling salesman problem (TSP). After calculating all pair-wise edge weights, the cost-optimal sequence of view poses is determined by means of Concorde [2], a fast TSP solver.

In order to define missions independent from a strictly structured warehouse model, an operator can define arbitrary 4D view poses in RViz using `interactive_markers`. A context menu at every marker allows to set a marker to the current MAV pose—this is especially useful to teach-in missions during manual flight—and to modify, add, or remove view poses.

7.2 Path Planning

The next layer when descending the planning hierarchy is a global path planner. This layer plans globally consistent plans, based on I) the SLAM-based environment model (as OctoMap), discretized to grid cells with 0.5 m edge length, II) the current pose estimate of the MAV as `nav_msgs/Odometry`, and

III) the next mission waypoint, including 3D position and yaw represented as `geometry_msgs/PoseStamped`. Planning frequency is 0.2 Hz and we use the A* algorithm to find cost-optimal paths.

In our application domain, most obstacles not represented in the allocentric map can be avoided locally, without the need for global replanning. Hence, it is sufficient to replan globally every five seconds to keep the local deviations of the planner synchronized to the global plan and to prevent the MAV from getting stuck in a local minimum that the local planner cannot escape due to its restricted view of the environment.

As via-points that are not mission critical can be blocked by locally perceived obstacles, it is not sufficient to send the next waypoint of the global path to the local planning layers. Instead, the input to the local planner is the complete global plan, which allows for skipping blocked via-points. The global path is cost-optimal with respect to the allocentric map. Hence, the path costs of the global path are a lower bound to path costs for plans refined based on newly acquired sensor information—mostly dynamic and static previously unknown obstacles. Locally shorter plans on lower layers with a local view on the map are not taken as they may yield globally suboptimal paths. Also, mission goals are not skipped as the local planner has to reach these exactly. If this is not possible, the mission planning has to resolve this failure condition.

7.3 Local Multiresolution Path Planning

On the local path planning layer, we employ a 3D local multiresolution path planner. This layer plans based on the allocentric path from the global path planner and local distance measurements which have been aggregated in a 3D local multiresolution map. It refines the global path according to the actual situation. The resulting more detailed trajectory is fed to the potential field-based reactive obstacle avoidance layer on the next level.

To resemble the relative accuracy of onboard sensors—i.e., they measure the vicinity of the robot more accurate and with higher density than distant space—we plan with a higher resolution close to the robot and with coarser resolutions with increasing distance.

Local multiresolution for path planning is also motivated by map dynamics. Since the parts of the plan that are farther away from the MAV are more likely to change, e.g., due to newly acquired sensor measurements, it is reasonable to spend more effort on a more detailed plan in the close vicinity of the robot. Compared to uniform resolution, our approach reduces planning time and makes frequent replanning feasible.

Our planner operates on grid-based robot-centric obstacle maps with higher resolution in the center and decreasing resolution in the distance. We embed an undirected graph into this grid and perform A* search from the center of the MAV-centered grid to the goal. The edge costs are given by the base obstacle costs of the cells it is connecting and its length given by the Euclidean distance between the cell centers.

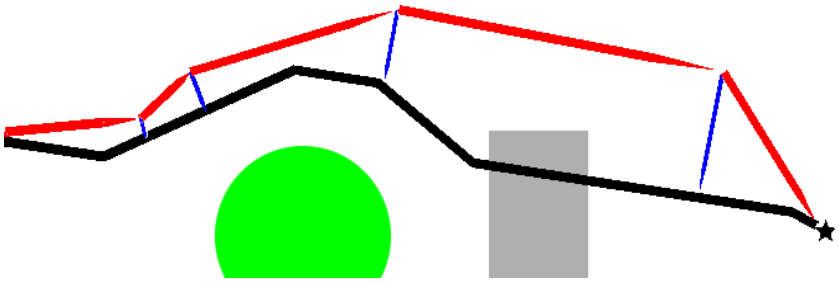


Fig. 18. The local plan (red) is coupled with the allocentric plan (black) by a cost term that penalizes deviations from the allocentric plan. The blue lines depict the deviation vectors at example points, the star is the planner's goal. The green circular obstacle is in the allocentric map, the gray rectangular obstacle has to be surrounded based on the local map.

An obstacle is modeled as a core with maximum costs, determined by obstacle radius r_F that is enlarged by the approximate robot radius r_R, and a distance-dependent part r_D that models the uncertainty of farther-away perceptions and motions with high costs. Added is a part with linearly decreasing costs with increasing distance to the obstacle r_S that the MAV shall avoid if possible. The integral of the obstacle stays constant by reducing its maximum costs h_{max} with increasing radius. For a distance d between a grid cell center and the obstacle center, the obstacle costs h_c are given by

$$h_c(d) = \begin{cases} h_{max} & \text{if } d \leq (r_F + r_D) \\ h_{max} \frac{1-d-(r_F+r_D)}{2*(r_F+r_D)} & \text{if } (r_F + r_D) < d < 3 * (r_F + r_D) \\ 0 & \text{otherwise} \end{cases} .$$

The local planner is coupled to the solution of the allocentric path planner by a cost term h_a, which is the shortest distance between a grid cell and any segment of the allocentric plan (see Fig. 18). The total cost h for traversing a grid cell is $h = w_1 \cdot h_c(d) + w_2 \cdot h_a$.

The output of the local navigation layer is the next waypoint along the planned path as `geometry_msgs/PoseStamped` in a robot-centric frame. This is further processed by a PID-controller to generate egocentric 4D velocity commands (v_x, v_y, v_z, v_{yaw}) published as `geometry_msgs/TwistStamped`. These commands are the input to the reactive obstacle avoidance layer.

7.4 Reactive Local Obstacle Avoidance

For safe navigation in complex environments, fast reliable obstacle avoidance is key. We developed a frequently updated local multiresolution obstacle map and a local reactive potential field-based collision avoidance layer to cope with dynamic and static obstacles. We transferred our previous work on obstacle

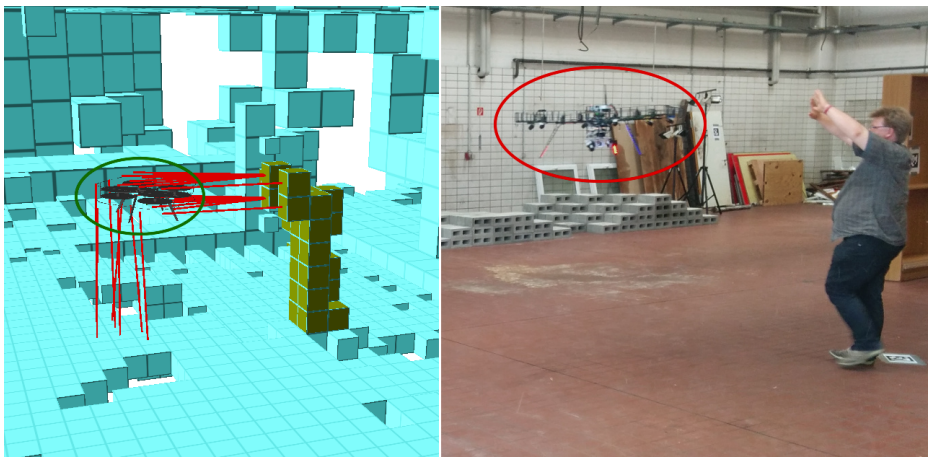


Fig. 19. The MAV is pushed away from an approaching person and the ground by potential field-based obstacle avoidance. Red lines in the left figure depict forces induced by the local obstacle map (cyan and yellow boxes, the yellow boxes depict the person) on the MAV.

perception and collision avoidance from our outdoor mapping MAV [4] to the system presented in this work.

To quickly react on obstacle perceptions, we use a version of the local multiresolution obstacle map (cf. Sec. 4.3) that is updated at the laser scanner frequency of 40 Hz. Obstacles represented in the map induce artificial repulsive forces to parts of the MAV, pushing it into free space. Fig. 19 shows an example, where the MAV avoids an approaching person and the ground. To take the MAV shape into account, we discretize it into 32 cells and apply the force to each cell. The resulting force vector and the velocity control vector from a higher navigation layer yield a velocity command that avoids obstacles, independent of localization. The obstacle avoidance layer runs at 20 Hz, equal to the frequency target velocities are sent to the low-level control layer. Velocity setpoints are published as `geometry_msgs/PoseStamped` and received from our ROS-MAVLink bridge node. The commands are sent to the Pixhawk Autopilot via the MAVLink protocol over a serial bus.

7.5 Velocity Control

Low-level velocity control is executed on the Pixhawk Autopilot, which receives 4D velocity setpoints via the MAVLink protocol. For linear velocity control, we use a modified Pixhawk Autopilot position control node. The node implements a PID-controller which calculates a 3D thrust vector based on the 3D linear velocity error. This leads to a 3D attitude and total thrust setpoint which is then used by lower-level controllers.

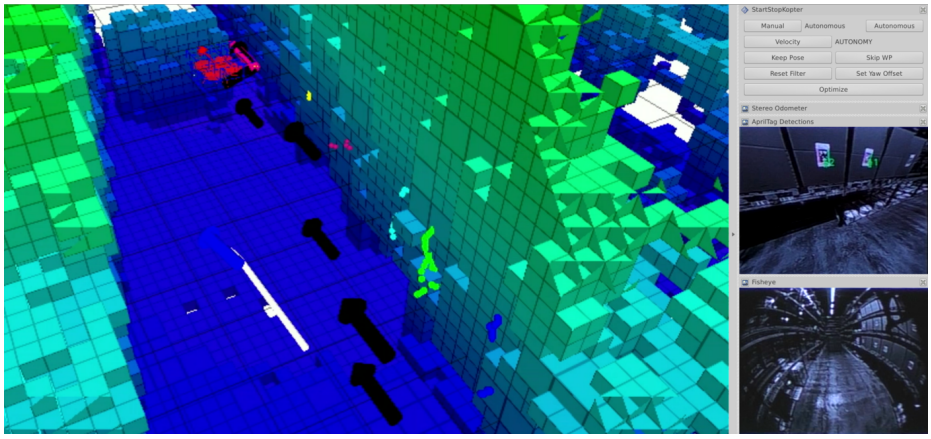


Fig. 20. Flight operator view. The RViz-based operator command and control interface depicts in the main window the allocentric obstacle map, the MAV pose (red shape), future mission waypoints (black arrows), obstacle-induced forces (not visible here since the MAV is sufficiently far away from obstacles), and 3D coordinates of detected AprilTags (clustered colored dots). Furthermore, approximate positions of RFID tags can be shown (not shown here). Other windows show visual AprilTag detections with corresponding ID in a rectified image (center-right) and the fisheye view from one front camera (bottom-right). An operator can choose between manual, velocity controlled, and fully autonomous operation. Quick commands—that have been identified as being especially useful during testing—include hovering at the current pose and skipping a waypoint.

We control the yaw Ψ of the MAV by a proportional controller $\Psi_{setp} = \Psi + K_p \cdot v_{yaw}$ with $K_p = 1$. Although the controller does not integrate the yaw rate v_{yaw} and thus shows a steady-state error when used open loop, it is well behaved in terms of steps in the resulting yaw setpoint Ψ_{setp}. Since we close the loop regarding yaw on a higher level, the described controller shows sufficient performance.

By limiting the maximum velocity setpoint received from the onboard computer to $2\,\frac{\mathrm{m}}{\mathrm{s}}$ in horizontal direction, $1\,\frac{\mathrm{m}}{\mathrm{s}}$ in vertical direction, and $0.2\,\frac{\mathrm{rad}}{\mathrm{s}}$ about yaw, even critical errors in ROS subsystems do not lead to severe effects at lower control layers on the MAV. We found these values to balance well between efficiency and safety in our application. Especially low yaw rates allow the safety pilot to intervene before the MAV rotates into an undesirable pose.

8 User Interfaces

8.1 Flight Operator Interfaces

Operating a complex robotic system in the field—especially for debugging and testing—requires a visualization of the system state easily monitored in real time

and the possibility to quickly send the most important commands to the system. These include, but are not limited to, switching between "manual", "velocity controlled", and "fully autonomous" operation. Furthermore, the operator has the ability to command the MAV to "fly to specific point" determined by an `interactive marker`, and "stay at current pose". The core visualization tool during flight is RViz, extended with several application-specific views/plugins. Typical views, possibly shown in parallel distributed to several computers, are:

- Allocentric view, showing mission and allocentric path planning, OctoMap, and localization (similar to Fig. 20),
- Egocentric view, showing local obstacle map, local path planning, and reactive obstacle avoidance (similar to Fig. 19 left),
- Localization view, showing SLAM-map, 3D laser scans, and visual odometry trajectories (similar to Fig. 14),
- Planning view, showing allocentric and egocentric path planning, and an overlay of allocentric and egocentric maps (similar to Fig. 17), and
- Vision view, showing camera images, tracked features, tracked AprilTags and visual odometry trajectories (similar to Fig. 21).

Whereas the allocentric and egocentric views are mainly used in field-testing and actual mission execution, the localization and planning views are more subsystem-specific and used during development and debugging. The vision view might be employed in both scenarios on demand.

Most nodes can be configured on-the-fly employing the `dynamic_reconfigure` framework. This is particularly important to parameterize lower-level systems, like the reactive obstacle avoidance and the camera system, but does also help to activate and deactivate features in high-level components.

The capabilities of ROS are not only used during a mission, but also during preparation and follow-up. As described in Sec. 5, we create an initial map by manually flying the MAV. During the manual flight, the MAV builds an allocentric map of the environment which is later used for localization and for defining a mission. An operator monitors the allocentric map using RViz to assure map coverage of the environment. We use an editing tool for post-processing the map[7]. The point cloud can be moved and rotated. Furthermore, specific points can be deleted and the whole point cloud can be aligned to a plane, e.g., the ground plane. We use this tool to align the origin of the map with the ground level and to orient the map north—an important prerequisite to maintain a common frame between laser pose tracking, IMU, and compass measurements.

In preparation of a mission, we can define missions by either creating a job list containing storage units and shelves to cover using an RViz plugin, shown in Fig. 17, or by manually defining 4D view poses employing `interactive_markers`. Furthermore, we use an interactive marker to set the initial pose of the MAV before takeoff for pose tracking.

Repeatability of experiments is important for efficient debugging and testing. Thus, we save user-defined missions (ordered set of 4D-waypoints) and can load

[7] Point cloud editor can be downloaded from `http://www.ais.uni-bonn.de/videos/ROS_book_2016`

them for consecutive experiments. Loading and editing those stored missions have turned out to significantly reduce the operator workload when testing the system, reducing the idle time of the system and resulting in a much higher possible test frequency.

We experienced that the time needed for preparation of a mission and/or adjusting parameters and fixing bugs, often exceeds the actual time needed for the flight itself. Furthermore, consecutive short flights with short landings in between are often possible without restarting onboard systems. Thus, to minimize the time for maintenance on ground, we do not restart the logging to Bag files. After successful mission execution, we use the ROS tool `MAV_bag_filter` to cut out Bag file segments containing individual flights and discard segments where the MAV status indicates that it is not flying. In this way, we are able to (a) significantly reduce the size of the Bag files and (b) minimize the amount of time needed for reviewing the data.

8.2 Safety Pilot Interfaces

While the flight operators monitor higher-level states of the MAV like proper initialization of the SLAM system and correct mission planning, we rely on a safety pilot to keep the MAV in a safe state during the whole mission. The safety pilot is able to monitor all status information that is vital for safe operation of the MAV in real time. This includes battery level, velocity setpoints, flight state, and many more. Incoming MAVLink packages from the Pixhawk Autopilot are encapsulated in a ROS message and sent to the ground control station over the wireless link. Here, the messages are extracted and streamed to the local network via UDP. For real-time visualization of the data streams, we employ the software QGroundcontrol[8]. Since this communication pipeline works bidirectionally, the safety pilot is also able to adjust flight parameters like, e.g., the maximum allowed vertical velocity during flight.

When an error occurs on a higher level or a subsystem fails, the safety pilot can always switch off the control authority of the onboard computer and recover the MAV. This can happen either with QGroundcontrol as well as with the manual remote control. We use this feature also during manual start and landing of the MAV. We manually start and land since we consider it to be safer than fully autonomous operation near the ground. By switching the control authority from manual mode to the onboard computer, we can totally eliminate the pilot in the loop. On the other hand, since we are able to completely switch off the autonomy, we can even deal with situations where the autonomy fails completely (e.g., if it should send velocity setpoints of NAN).

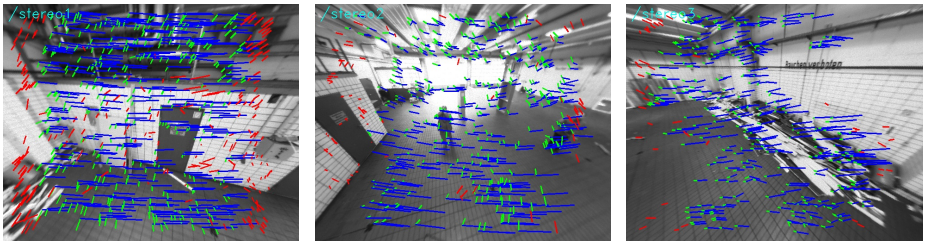
9 Experiments and Evaluation

We evaluated the individual components of our MAV in simulation and flight experiments in our lab. Furthermore, the integrated system was tested and demon-

[8] `http://qgroundcontrol.com`

Table 4. Camera frame rate is limited by exposure time.

Exposure time (ms)	Frame rate (Hz)
40	17
23	25
17	30
3	50



(a) Front stereo pair (b) Back-left stereo pair (c) Back-right stereo pair

Fig. 21. Visualization of the correspondences in the three stereo camera pairs. Correspondences between one stereo pair are colored blue. Feature correspondences tracked by `viso2` are colored green (inliers) and red (outliers).

strated in a warehouse of a logistics company to achieve a realistic test environment. In addition to the evaluation results, we report lessons learned during development and testing of the system.

9.1 Data Acquisition

Fig. 7 shows point clouds recorded with the 3D laser scanner. Due to the different angular mounting of the 2D laser scanners (cf. Fig. 6), we minimize the blind spots in the vicinity of the MAV. Occlusions, e.g., caused by the frame or propellers occur in different directions and can be compensated by measurements from different poses. This results in an omnidirectional FoV with a minimal blind spot.

We estimated the accuracy of the Hokuyo UST-20LX and compared it to the Hokuyo UTM-30LX-EW used in our previous work [4]. Indoors, both laser scanners show the same accuracy of $\sim \pm 10\,\text{mm}$ when measuring a 0.5 m distant object. Outdoors, the accuracy of the UTM-30LX-EW stays the same, but the accuracy of the Hokuyo UST-20LX degrades to $\sim \pm 35\,\text{mm}$.

We evaluated the data acquisition speed of the synchronized cameras. Although the maximum frame rate is up to 55 fps, it is limited by the exposure time of the cameras. Tab. 4 reports the resulting frame rates.

Fig. 21 shows a typical image set, captured during flight. It can be seen that the visual odometry finds most correspondences correctly, but some false cor-

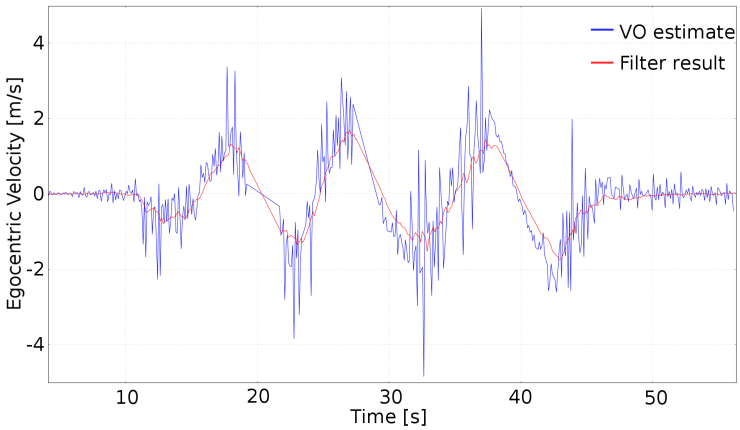


Fig. 22. Egocentric velocity estimate from visual odometry in forward direction and filter result.

respondences are produced due to repetitive environment structures and strong illumination differences. Nevertheless, due to the redundant structure and the correspondence-dependent weighting, the visual odometry does not lose track, even if one instance finds no correspondence at all. The computation time for visual odometry including image rectification is 30 ms per stereo camera image pair.

In order to evaluate the robustness of the filter, we measured the visual odometry velocity while flying a sinusoidal trajectory. Only accelerometer, gyroscopes, and one visual odometry estimate are used to correct the filter. Fig. 22 shows the visual odometry input and the filter result. Although the visual odometry loses track (at $t = 19\,\mathrm{s}$ and $t = 28\,\mathrm{s}$), the filter is able to bridge this information gap. In normal operation, this gap would also be filled by other velocity estimates.

9.2 RFID Detection

RFID tags are detected and mapped in the allocentric map. Instead of using an elaborated sensor model, we approximate the tag positions by a predefined offset from the RFID antenna. This is sufficient to match tag readings to storage places. For our specific case, we found an offset of 0.5 m to be appropriate. Fig. 23 displays the detected tags, mounted on individual storage places during a mission in the warehouse depicted in Fig. 11. It can be seen that the achievable accuracy lies within the dimensions of one storage unit, capable of storing one EUR-pallet of size $80 \times 120\,\mathrm{cm}$. Therefore our system is capable of performing a per-storage-unit attribution of EUR-pallets.

9.3 Flight Time

We evaluated our system in flight experiments. When manually flying the MAV indoors, we measured a flight time between 6 min to 8 min, depending on the

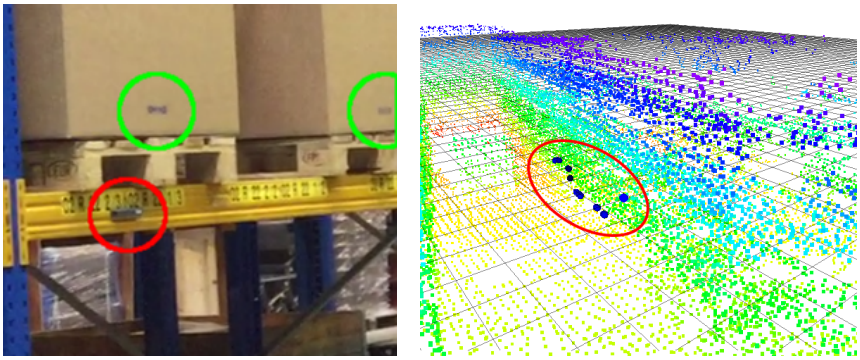


Fig. 23. RFID detections in a warehouse aisle. We map RFID tags during flight with the current MAV pose and a predefined scanning distance. Left: Photo of the scene. Each storage unit (red circle) and every stock (green circle) is marked with RFID Tags. Right: Representation of the scene in RViz. The detections are depicted as blue spheres.

Table 5. MAV components emitting and/or receiving radio waves.

Component	Frequency (GHz)
GPS L1	1.57542
GPS L2	1.2276
GLONASS L1	1.6
Computer CPU	0.8 – 3.2
Computer memory	1.6
WiFi	5.15 – 5.725
Remote Control	2.4
RFID UHF	0.865 – 0.869
	0.902 – 0.928

flight dynamics. This is sufficient for typical indoor inspection tasks (described in detail in Sec. 9.6) and to scan one typical warehouse aisle with ∼50 m length and ∼5 m height with an average horizontal velocity of ∼0.5 $\frac{\text{m}}{\text{s}}$. Furthermore, the ability to hot-swap batteries compensates for the relatively short flight time.

9.4 Electromagnetic Compatibility

Several components on the MAV emit radio waves. We evaluated the influence of these components on each other by identifying the relevant frequencies in a series of tests. Tab. 5 gives an overview on the components and frequencies. Although it does not show the exact emission spectrum, it provides initial information which frequency ranges are prone to interference for further investigation.

Although our system is primarily built to work in GNSS-denied environments, our MAV is equipped with an optional GNSS antenna for use in external stock. It can be seen that the computer memory is working at the same clock frequency as the GNSS sources. We found that it emits interference radiation preventing a stable GNSS reception. Since we experienced strong interference especially with GPS, the GNSS antenna was placed as far as possible from the

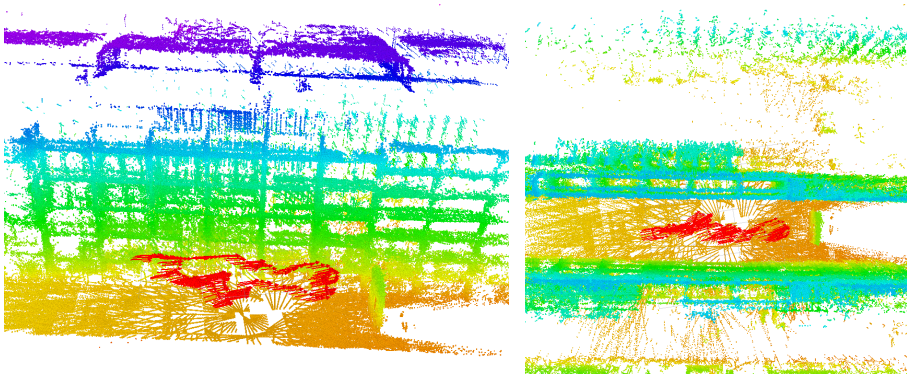


Fig. 24. Localization result. The MAV trajectory (red arrows) is tracked by means of laser scan registration, combined with visual odometry and IMU measurements. This yields 6D pose estimates. Shown is a flight through a warehouse aisle. In the side-view (left), the relation to the accurately mapped storage units can be seen. In the top-down view (right), it can be seen that the pose is tracked despite considerable self-similarity of the shelves. Map color encodes height.

jamming source to reduce noise, which yielded sufficient reception of the GPS signal. We did not experience other noteworthy interferences.

Benchmarking the WiFi network gives a real throughput of 7.5 MB/s. Latency analysis gives an average ping of 1.22 ms ± 0.11 ms. We aim for a fully autonomous system, so no data has to be exchanged between the ground control stations and the MAV in normal operation modes, except for a mission specification before takeoff and data transfer to the ground station after landing. This benchmark shows that the communication infrastructure enables the operators to visualize point clouds or even view live video feeds with ∼ 2 Hz for debugging purposes.

9.5 Mapping and Pose Tracking

We performed experiments with the integrated system. Fig. 24 shows the resulting trajectory of our indoor localization experiment. We build a map with the onboard laser sensors before mission start. During a mission, the 3D laser scans—aggregated over 500 ms—are registered to the map yielding a 6D pose estimate at 2 Hz. The resulting trajectory is globally consistent.

In order to assess the performance of our global registration and allocentric mapping approach, we tested our method on a dataset of the parking garage[9]. Without pose graph optimization, the trajectory aggregates drift which results in inconsistencies, indicated by a misalignment of the walls. Our registration method with graph optimization yields accurate results. Fig. 25 shows details of

[9] Datasets recorded in-flight with an MAV are available at: `http://www.ais.uni-bonn.de/mav_mapping`.

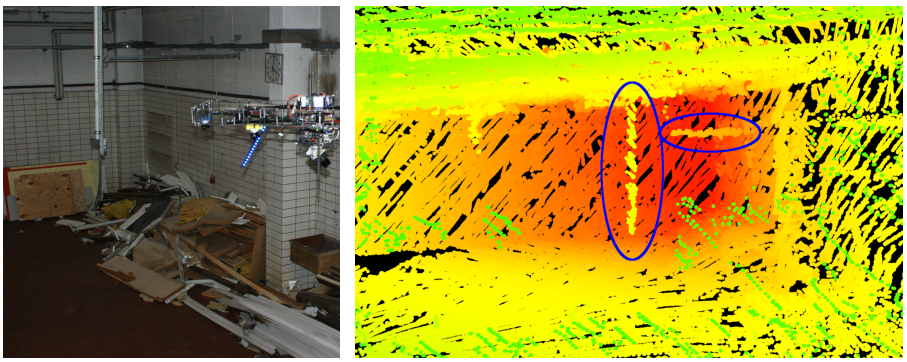


Fig. 25. Impressions of the quality of the built 3D map. Environmental structures are consistently mapped. Even details such as the narrow pipe structure and a cable canal (circled) are accurately modeled. Color encodes the distance to the view-point.

a map of a garage environment. Here, even narrow structures like pipes can be identified in the globally aligned 3D scans. For a detailed comparison with other registration methods see [4].

9.6 Navigation

We evaluated the autonomous navigation by flying missions in a warehouse. The MAV visits several manually defined observation poses on different heights along a shelf based on an allocentric map created with our SLAM approach. Fig. 20 shows one example mission. The MAV successfully accomplished multiple missions with a duration between ∼2 min to 5 min and a total trajectory length of ∼40 m to 80 m each.

To evaluate the local obstacle avoidance, we control the MAV with egocentric velocity commands, i.e., a zero velocity setpoint for movements in the plane and rotations, and a small descent velocity to keep the MAV close to the ground. The obstacle avoidance keeps the MAV at a safe distance to the ground. Fig. 19 shows an experiment where a person approaches the MAV. The MAV avoided all static and dynamic obstacles based on the 3D laser scans.

A video showing autonomous mission execution and reactive obstacle avoidance can be found on our website[10].

10 Lessons Learned

The use of ROS was extremely valuable for development and evaluation of the described MAV system. We experienced the MAV to be a very complex mechatronic system consisting of many individual hard- and software subsystems. Most

[10] `http://www.ais.uni-bonn.de/videos/ROS_book_2016`

subsystems offer no redundancy and show a Single Point of Failure (SPOF) characteristic.

Due to the modular and transparent ROS framework, development and error treatment was greatly simplified. Since the publish–subscribe pattern offers transparency, the effort for error analysis was reduced to a minimum. Logging of data to Bag files further simplifies error analysis.

The modular design and abstraction to ROS nodes facilitates the fast development of software and allowed us to transfer technology between multiple MAVs and even ground robots. Since the MAV is connected to several external hard- and software components, the loose coupling via ROS messages massively simplifies the integration effort. Furthermore, we use standard message formats shipped with ROS and relied on third-party modules whenever possible. This facilitates both replacing submodules of the system with modules developed for other robots or even in other research groups with often low adaptation effort and the maximum use of already available ROS debugging and visualization tools.

Since ROS handles the transportation of messages, effortful data routing between the MAV and ground stations as described in Sec. 8 is not required. Nevertheless, since ROS is not real-time capable, we advice to use the `tcpNoDelay` transport hint for nodes that are crucial for real-time control. We use the no delay transport hint, e.g., regarding all communication with the Pixhawk autopilot.

Real-time visualization of data streams, especially camera images, laser scans, and planned trajectories with, e.g., RViz, and `rqt_plot`, made it possible to develop such a complex system. Real-time adjustment of crucial parameters like, e.g., camera exposure time, using `dynamic_reconfigure` sped up the development phase and also helped during evaluation.

During development of inherently unstable SPOF systems we made extensive use of simulation technology like, e.g., Gazebo, where failures are permitted.

Development of sophisticated software modules for, e.g., state estimation or action planning, was facilitated by the extensive software library which is already shipped with ROS. We rely on many standard components like drivers (e.g., `urg_node` for the laser scanners) that otherwise would be costly to develop.

Representing a complex mechatronic system in software benefited from tools like `tf` and `robot_state_publisher`. The kinematic tree represents not only statically calibrated nodes like `base_link` ⇔ `camera_1..6`, but also dynamic relations like `base_link` ⇔ `laser_scanner` or `base_link` ⇔ `map_origin`. By using the above mentioned ROS packages, we avert the cumbersome and error prone manual track keeping of a variety of multidimensional transformations. We want to advice here that although `tf` offers a transparent way to maintain transformations, to always check the tf tree for consistency with tools like `view_frames`.

We make extensive use of `screen` when operating the MAV. It proved to be very useful to start the `roscore` and, when working with multiple operators, all additional components in a respective screen session. Thus, if the WiFi connec-

tion drops, all components are easily recoverable and screens can be exchanged between operators.

When operating a complex robotic system in the field, it is inevitable to have well-organized processes and a tested hardware setup to not waste valuable testing time on site. In particular, clear responsibilities are important, e.g., who starts which subsystems and is responsible for their configurations—including parameter checking before takeoff and monitoring during flight. Furthermore, sufficient attention must be given to important infrastructure, like reliable networking and WiFi connections, standardized software setups on workstations, wiring of all operator station components, and if applicable, a directly available contact person on site to organize important prerequisites as power or networking and solve problems in a timely manner. The above mentioned precautions facilitate efficient usage of testing time and maximize the benefits of operations in the field.

11 Conclusions

In this chapter, we presented a cognitive MAV that is capable of semantically perceiving its environment and planning inventory missions.

We approached this challenge by employing a multimodal omnidirectional sensor setup to achieve situation awareness. The sensors have a high data rate for tracking the MAV motion and for quick detection of changes in its environment.

Our ROS-based mapping and navigation pipeline allows for fully autonomous flight even in GNSS-denied environments.

Ample onboard processing power in combination with a high bandwidth ground connection leads to a system that is suitable to deploy and debug custom algorithms and for conducting further research. The ability to hot-swap batteries and/or ground power supply makes developing and testing highly efficient.

We showed the system robustness in multiple indoor experiments where the only manual interactions were the starting and landing phases. Thus, the system is able to inspect areas in a fully autonomous mission.

Author's Biographies

Marius Beul received his M.Sc. degree in Electrical Engineering in 2013 from Cologne University of Applied Sciences. Since October 2013, he works as a member of the scientific staff in the Autonomous Intelligent Systems Group at the University of Bonn. His research interests include Aerial Robotics, State Estimation, Path Planning and Control.

Nicola Krombach obtained her M.Sc. degree in Computer Science from Rheinische Friedrich-Wilhelms Universität Bonn in 2016. Since March 2016, she is a researcher in the Autonomous Intelligent Systems Group at the University of Bonn. Her research interests include image processing and visual SLAM.

Matthias Nieuwenhuisen received his Diploma in Computer Science from Rheinische Friedrich-Wilhelms Universität Bonn in 2009. Since May 2009, he is a researcher in the Autonomous Intelligent Systems Group at the University of Bonn. His current research interests include path and motion planning for MAVs.

David Droeschel received a M.Sc. degree in Autonomous Systems from the University of Applied Sciences Bonn-Rhein-Sieg in 2009. Since May 2009, he is a researcher in the Autonomous Intelligent Systems Group of the University of Bonn. His research interests include efficient 3D perception and SLAM.

Sven Behnke received his Diploma in Computer Science from Martin-Luther-Universität Halle-Wittenberg in 1997 and Ph.D. from Freie Universität Berlin in 2002. He worked in 2003 as postdoctoral researcher at the International Computer Science Institute, Berkeley. From 2004 to 2008, he headed the Humanoid Robots Group at Albert-Ludwigs-Universität Freiburg. Since 2008, he is professor for Autonomous Intelligent Systems at the University of Bonn. His research interests include cognitive robotics, computer vision, and machine learning.

References

1. Abraham, S., Förstner, W.: Fish-eye-stereo calibration and epipolar rectification. ISPRS Journal of Photogrammetry and Remote Sensing 59(5), 278–288 (2005)
2. Applegate, D., Bixby, R., Chvatal, V., Cook, W.: Concorde TSP solver (2006)
3. Chambers, A., Achar, S., Nuske, S., Rehder, J., Kitt, B., Chamberlain, L., Haines, J., Scherer, S., , Singh, S.: Perception for a river mapping robot. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (2011)
4. Droeschel, D., Nieuwenhuisen, M., Beul, M., Holz, D., Stückler, J., Behnke, S.: Multilayered mapping and navigation for autonomous micro aerial vehicles. Journal of Field Robotics 33, 451–475 (2016)
5. Droeschel, D., Stückler, J., Behnke, S.: Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2014)
6. Droeschel, D., Stückler, J., Behnke, S.: Local multi-resolution surfel grids for MAV motion estimation and 3D mapping. In: Proc. of the Int. Conf. on Intelligent Autonomous Systems (IAS) (2014)
7. Fiedler, M.: Inventairy. `http://www.inventairy.de/` (2016), [german]
8. Flores, G., Zhou, S., Lozano, R., Castillo, P.: A vision and GPS-based real-time trajectory planning for a MAV in unknown and low-sunlight environments. Journal of Intelligent & Robotic Systems 74(1-2), 59–67 (2014)
9. Fossel, J., Hennes, D., Claes, D., Alers, S., Tuyls, K.: OctoSLAM: A 3D mapping approach to situational awareness of unmanned aerial vehicles. In: Proc. of the Int. Conf. on Unmanned Aircraft Systems (ICUAS) (2013)
10. Geiger, A., Ziegler, J., Stiller, C.: StereoScan: Dense 3D reconstruction in real-time. In: IEEE Intelligent Vehicles Symposium (2011)
11. Grzonka, S., Grisetti, G., Burgard, W.: A fully autonomous indoor quadrotor. IEEE Trans. on Robotics 28(1), 90–100 (2012)

12. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots 34, 189–206 (2013)
13. Huh, S., Shim, D., Kim, J.: Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of UAVs. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (2013)
14. Jutzi, B., Weinmann, M., Meidow, J.: Weighted data fusion for UAV-borne 3D mapping with camera and line laser scanner. International Journal of Image and Data Fusion (2014)
15. Kohlbrecher, S., Meyer, J., von Stryk, O., Klingauf, U.: A flexible and scalable SLAM system with full 3D motion estimation. In: Proc. of the IEEE Int. Symposium on Safety, Security and Rescue Robotics (SSRR) (2011)
16. Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: A general framework for graph optimization. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 3607–3613 (2011)
17. Magree, D., Mooney, J.G., Johnson, E.N.: Monocular visual mapping for obstacle avoidance on UAVs. Journal of Intelligent & Robotic Systems 74(1-2), 17–26 (2014)
18. Meier, L.: Micro aerial vehicle link protocol (MAVLink). `mavlink.org` (2015)
19. Meier, L., Tanskanen, P., Heng, L., Lee, G., Fraundorfer, F., Pollefeys, M.: PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. Autonomous Robots 33(1-2), 21–39 (2012)
20. Moore, R., Dantu, K., Barrows, G., Nagpal, R.: Autonomous MAV guidance with a lightweight omnidirectional vision sensor. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2014)
21. Mori, T., Scherer, S.: First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2013)
22. Morris, W., Dryanovski, I., Xiao, J., Member, S.: 3D indoor mapping for micro-UAVs using hybrid range finders and multi-volume occupancy grids. In: In RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras (2010)
23. Olson, E.: AprilTag: A robust and flexible visual fiducial system. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2011)
24. Park, J., Kim, Y.: 3D shape mapping of obstacle using stereo vision sensor on quadrotor UAV. In: AIAA Guidance, Navigation, and Control Conference (2014)
25. Pons, J.: DroneScan - Airborne Data Collection. `http://www.dronescan.co/` (2016)
26. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive uav control in cluttered natural environments. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2013)
27. Schadler, M., Stückler, J., Behnke, S.: Multi-resolution surfel mapping and real-time pose tracking using a continuously rotating 2D laser scanner. In: Proc. of the IEEE Int. Symposium on Safety, Security and Rescue Robotics (SSRR) (2013)
28. Schauwecker, K., Zell, A.: On-board dual-stereo-vision for the navigation of an autonomous MAV. Journal of Intelligent & Robotic Systems 74(1-2), 1–16 (2014)
29. Schmid, K., Lutz, P., Tomic, T., Mair, E., Hirschmüller, H.: Autonomous vision-based micro air vehicle for indoor and outdoor navigation. Journal of Field Robotics 31(4), 537–570 (2014)
30. Schwarz, M., Beul, M., Droeschel, D., Schüller, S., Periyasamy, A.S., Lenz, C., Schreiber, M., Behnke, S.: Supervised autonomy for exploration and mobile ma-

nipulation in rough terrain with a centaur-like robot. Frontiers in Robotics and AI, section Humanoid Robotics (2016)
31. Stückler, J., Behnke, S.: Multi-resolution surfel maps for efficient dense 3D modeling and tracking. Journal of Visual Communication and Image Representation 25(1), 137–147 (2014)
32. Tomić, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I., Ruess, F., Suppa, M., Burschka, D.: Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. Robotics Automation Magazine, IEEE 19(3), 46–56 (2012)
33. Tripathi, A., G Raja, R., Padhi, R.: Reactive collision avoidance of UAVs with stereovision camera sensors using UKF. In: Advances in Control and Optimization of Dynamical Systems. pp. 1119–1125 (2014)