Received: 15 May 2019 | Revised: 16 November 2019 | Accepted: 16 February 2020

DOI: 10.1002/rob.21949

SYSTEMS ARTICLE

WILEY

ARDEA—An MAV with skills for future planetary missions

Philipp Lutz[1] | Marcus G. Müller[1] | Moritz Maier[1] | Samantha Stoneman[1] | Teodor Tomić[2] | Ingo von Bargen[1] | Martin J. Schuster[1] | Florian Steidle[1] | Armin Wedler[1] | Wolfgang Stürzl[1] | Rudolph Triebel[1]

[1]Robotics and Mechatronics Center (RMC), German Aerospace Center (DLR), Weßling, Germany

[2]Skydio and Munich School of Robotics and Machine Intelligence, Technical University of Munich (TUM), Munich, Germany

Correspondence
Philipp Lutz, Robotics and Mechatronics Center (RMC), German Aerospace Center (DLR), Germany.
Email: Philipp.Lutz@dlr.de

Funding information
Helmholtz-Gemeinschaft, Grant/Award Numbers: ARCHES/ZT-0033, ROBEX/HA-304; Horizon 2020 Framework Programme, Grant/Award Number: AUTOPILOT/731993

Abstract

We introduce a prototype flying platform for planetary exploration: autonomous robot design for extraterrestrial applications (ARDEA). Communication with unmanned missions beyond Earth orbit suffers from time delay, thus a key criterion for robotic exploration is a robot's ability to perform tasks without human intervention. For autonomous operation, all computations should be done on-board and Global Navigation Satellite System (GNSS) should not be relied on for navigation purposes. Given these objectives ARDEA is equipped with two pairs of wide-angle stereo cameras and an inertial measurement unit (IMU) for robust visual-inertial navigation and time-efficient, omni-directional 3D mapping. The four cameras cover a 240° vertical field of view, enabling the system to operate in confined environments such as caves formed by lava tubes. The captured images are split into several pinhole cameras, which are used for simultaneously running visual odometries. The stereo output is used for simultaneous localization and mapping, 3D map generation and collision-free motion planning. To operate the vehicle efficiently for a variety of missions, ARDEA's capabilities have been modularized into skills which can be assembled to fulfill a mission's objectives. These skills are defined generically so that they are independent of the robot configuration, making the approach suitable for different heterogeneous robotic teams. The diverse skill set also makes the micro aerial vehicle (MAV) useful for any task where autonomous exploration is needed. For example terrestrial search and rescue missions where visual navigation in GNSS-denied indoor environments is crucial, such as partially collapsed man-made structures like buildings or tunnels. We have demonstrated the robustness of our system in indoor and outdoor field tests.

KEYWORDS

aerial robotics, computer vision, exploration, GPS-denied operation, planetary robotics

Philipp Lutz and Marcus G. Muller contributed equally to this work.

1 | INTRODUCTION

In recent years, MAVs have experienced an increased attention in the consumer market, in industrial applications and also in robotics research. The consumer market is currently focused on products for aerial photography, competitive race flying and autonomous "follow-me" video operation for sports activities. Competitive flying has become so popular that it has evolved into a professional discipline with a dedicated league, the Drone Racing League (DRL)[1]. Recent technology in simultaneous localization and mapping (SLAM) and volumetric mapping have additionally paved the way for MAVs to be used in virtual reality (VR) and augmented reality (AR) applications, where an important feature of such systems is a stereo camera rig with a large field of view (FOV).

Furthermore, MAVs have also become of interest for inspection and maintenance tasks in industrial applications. Example tasks include the detection of leaks in gas pipelines, or cracks in bridges, as well as the volumetric mapping of construction sites or areas of world heritage for documentation and further analysis. In search and rescue (SAR) scenarios MAVs can help to identify people in need, provide aid from above and build a disaster map for emergency response teams. Recently, MAVs have even been considered useful in production lines of the "Factory of the Future" (Augugliaro et al., 2014). A common requirement for all of the aforementioned applications is a means of visual sensing and a GNSS device that is used for global localization and navigation.

Finally, MAVs are also being developed for autonomous exploration and mapping of extraterrestrial bodies (Huber, 2016). They are ideally suited for scouting purposes, since they can quickly cover large areas of interest and reach places that are inaccessible for ground-based robots such as rovers. Planetary scientists have high expectations that flying vehicles could be applied for autonomous exploration and mapping of relevant areas such as lava tubes on Mars (Daga et al., 2009). Depending on whether there is an atmosphere or not, a classical propeller propulsion system or booster engines can be used interchangeably on an MAV platform, with only minor changes in the navigation software.

However, in addition to the specific hardware requirements for MAVs in space applications, there are two major challenges for the navigation software. First, in contrast to most terrestrial applications, no GNSS device can be used, which is why we rely on cameras and an IMU as sensor modalities. Second, the required level of on-board autonomy is much higher. While most commercially available MAVs are only partly autonomous, that is they are either directly controlled by a human operator, or the operator must be able to intervene at any time, unmanned space missions usually have communication round-trip times of more than a minute. For instance, the round-trip time for a signal from Mars to Earth and back is between 8 and 40 min (Mankins, 1987), depending on their constellation. This is too long for any intervention by a human operator and can render the exploring vehicle out of operation. Compared to passive exteroceptive sensors, for example, cameras, many active exteroceptive sensors, such as Radio Detection and Ranging (RADAR) and Light Detection and Ranging (LIDAR), are characterized by high energy consumption, heavy weight, and more difficult space qualification. Cameras are lightweight and capture an information-rich representation of the environment, which makes them ideally suited for mobile robots that have limited payload. In addition to navigation purposes, cameras can be used for higher level mission tasks, such as scientific inspection of the environment or even taking selfies. This was done by the Curiosity rover team (Maki et al., 2012) and provided a convenient means of inspecting the rover itself. Stereo cameras have been successfully employed on MAVs to obtain depth information in both indoor and outdoor environments (Barry & Tedrake, 2015; Gohl, Honegger, Omari, Achtelik, & Siegwart, 2015; Matthies, Brockers, Kuwata, & Weiss, 2014; M. G. Müller et al., 2018; Tomić et al., 2012). In the past these cameras have been space-tested and used in several planetary robotic systems (Maimone, Johnson, Cheng, Willson, & Matthies, 2006).

To solve complex scientific investigation and exploration tasks in an effective and efficient manner, a team of heterogeneous robots can be used to distribute specific tasks to specialized team members. Moreover, crucial skills can be distributed across multiple members of the team to reduce the danger of a single point of failure. Reusing modular software and hardware components across all systems additionally reduces the complexity and effort in designing a robotic team.

This motivated us to build ARDEA, shown in Figure 1. The MAV supports wide-angle stereo vision, runs all computations onboard and performs navigation functions autonomously. ARDEA was not built for one specific task in mind, rather its set of parameterizable skills can be used to assemble complex missions. The human operator or the robotic team can choose skills to perform a specific task and accomplish the overall mission. The skills are defined generically enough to be applicable to other robots in the team. Also, a skill should be intuitive to use and work robustly, so that even unexperienced operators can assemble new missions in a fast and efficient manner. Together with our lightweight rover unit

FIGURE 1 In-house built hexacopter autonomous robot design for extraterrestrial application on Mt. Etna. Its size and weight is 68 × 68 × 30 cm and 2.65 kg, including battery, respectively [Color figure can be viewed at wileyonlinelibrary.com]

[1]https://thedroneracingleague.com

(LRU; Schuster et al., 2017), which is equipped with a landing platform, the presented MAV forms a heterogeneous robot team. The core navigation software components, such as visual odometry (VO), local reference filtering and SLAM are designed to operate on both systems, differing only in configuration. We emphasize that the design and development of space-qualified hardware is beyond the scope of this paper. Instead, we focus on the algorithmic design of visual-inertial navigation for MAVs. Furthermore, while the specific platform we present here requires an atmosphere for flying, we note that our navigation system can be used similarly on hovering vehicles with thrusters or even on ground rovers or underwater vehicles.

First, we discuss the current state of planetary exploration using MAVs, existing design concepts and state of the art navigation algorithms in Section 2. In Section 3, we describe general hardware and software design considerations and the resulting setup of ARDEA. Next, we present low and high level autonomy software components in Section 4. In Section 5 basic MAV skills, the building blocks of missions, are described. After defining the system design and skills, we demonstrate its capabilities in indoor and outdoor field experiments in Section 6. Finally, Section 7 gives concluding remarks and addresses potential future work.

2 | RELATED WORK

A large body of literature is concerned with sensors and autonomy functions of MAVs. We begin with a discussion on research about building flying robots for planetary exploration. Contemporary works dealing with MAVs with a design similar to that of ARDEA are then presented. We then give a brief overview of the literature regarding each of ARDEA's crucial components: visual-inertial navigation, motion planning, and control.

2.1 | Robotic planetary exploration with MAVs

In the community of robotic exploration the idea of using some sort of MAV has emerged over the last years. Future rotor-based robotic vehicles will be able to fly on planets and moons with a sufficiently dense atmosphere, such as Mars (Huber, 2016) or Saturn's moon Titan (Lorenz et al., 2017). Such lightweight flying robots are envisioned to be able to travel distances of up to 5 km (Thangavelautham et al., 2014), which is enough to gain an overview and aid the navigation on accessible terrain of slower ground rovers that can carry heavier payloads and manipulators. The Jet Propulsion Laboratory plans on sending a small coaxial copter to Mars in the *Mars2020* mission (Balaram et al., 2018) for scouting. Also the *Global Exploration Roadmap* (ISECG, 2018) states that the exploration of Martian lava tubes is of high scientific interest. This raises challenging requirements for robots which are capable of mapping these difficult to reach geological points of interest (POIs). An obvious choice for those requirements are propeller and rotor aircraft designs, however, also other types like fixed wings were proposed for planetary mission, as shown in Kuhl (2008). Although these approaches are well suited to cover vast areas efficiently, they are infeasible for narrow cave exploration. In addition, the presented system needs a large starting and landing area and is powered by a propulsion system which has to be refueled. Another approach is to send out robotic flapping wing fliers of a bumblebee size (Kang et al., 2019) along with a ground rover unit to do collaborative exploration. All of the discussed designs are restricted to celestial bodies with an atmosphere. Without an atmosphere the flying robots would need a different propulsion system, such as thruster-based propulsion, which is out of scope for our research.

2.2 | Autonomous MAV system designs

Compared to other aerial vehicles such as helicopters, multirotor systems have simple mechanical designs which are highly customizable. The most common designs are based on quadrotor platforms (Schmid, Lutz, Tomić, Mair, & Hirschmüller, 2014; Tomić et al., 2012) with different rotor configurations. Depending on the application, for example, industrial inspection, surveillance, SAR or planetary exploration, different designs might be more suitable. The first design aspect is the MAV size and it is defined by the narrowest traversable operation space. This can be simply addressed by constraining the size of the vehicle or by changing the shape of the frame on the fly. Those designs have one additional degrees of freedom (DOF) per frame arm and have been shown to either change the frame shape during flight in an adaptive morphology way (Falanga, Kleber, Mintchev, Floreano, & Scaramuzza, 2019) or by steering the motor thrust vectors in a way that the MAV can fly and hover in arbitrary orientations (Kamel et al., 2018) and therefore, pass narrow passages.

Another design aspect is concerned with safety in case a motor failure. Fail-safe robustness is achieved by adding redundancy to the propulsion system and failure detection and handling in the control software. In Michieletto, Ryll, and Franchi, 2018 and M. Müller and D'Andrea (2014, 2016) suitable system designs for motor redundancy and necessary control strategies to handle motor faults are discussed. We discuss this aspect of our system design in Section 4.1.1.

Sensor placement is a major design aspect resulting again from the operation environment properties. In literature most MAVs which are suitable for operating in indoor and outdoor environments are using cameras as their main sensor. They either use one monocular camera (M. Achtelik, Achtelik, Weiss, & Siegwart, 2011; Ok, Gamage, Drummond, Dellaert, & Roy, 2015; Weiss, Achtelik, Lynen, Chli, & Siegwart, 2012), a stereo setup (Matthies et al., 2014; Schmid, Lutz, et al., 2014; Tomić et al., 2012) or even multiple stereo setups (Schauwecker & Zell, 2014). To further enhance the FOV, (Schneider & Förstner, 2015) used a wide angle stereo camera configuration. Some approaches combine several exteroceptive sensors such as cameras and LIDARs. For instance, in Beul, Krombach, Nieuwenhuisen, Droeschel, and Behnke (2017) two 3D laser scanners are combined with three stereo camera pairs to achieve a larger

FOV for confined spaces such as warehouses. Our hardware design was mainly driven by having the widest possible unobstructed stereo camera FOV setup.

2.3 | Visual-inertial navigation

Visual-inertial navigation has received a great amount of attention in the last decades and several approaches have been suggested. These navigation approaches are crucial for a flying system, which cannot rely on GNSS. They can be roughly categorized into filter- and optimization-based approaches. While optimization-based approaches can have advantages in terms of accuracy (Forster, Carlone, Dellaert, & Scaramuzza, 2017) by relinearizing the estimated state, they must solve challenges arising from the high frequency of inertial measurements. Filter-based approaches can be used to develop solutions that have lower demands on computing capacities and yet achieve comparable accuracies (Mourikis & Roumeliotis, 2007; Schmid, Ruess, & Burschka, 2014). Most visual-inertial odometries (VIOs) for MAVs use an IMU in combination with a single camera (Bloesch, Omari, Hutter, & Siegwart, 2015; Leutenegger, Lynen, Bosse, Siegwart, & Furgale, 2015; Mourikis & Roumeliotis, 2007; Qin, Li, & Shen, 2018). An in-depth evaluation of a number of popular VIOs available as open-source software can be found in Delmerico and Scaramuzza (2018).

Advantage of a stereo odometry is that it can estimate scale directly from visual input, at least for close range flight. On our system we chose to have depth from high-quality semiglobal matching (SGM) stereo provided by an external field programmable gate array (FPGA) board and estimate motion by means of 3D point alignment, which is fast and can be performed in closed form, see Section 4.2.1. There are only few approaches that employ more than two cameras. In Houben, Quenzel, Krombach, and Behnke (2016), the visual features from multiple cameras are tracked in a joint optimization, leading to a tight coupling. In contrast, we process our stereo pairs decoupled form each other, similar to Beul et al. (2015), and fuse their VO results in a real-time capable filter for local state estimation. Thereby, the complexity of our global graph-based estimation is not affected by the number of high-frequency sensors, allowing for fast online optimization steps (Schuster, Schmid, Brand, & Beetz, 2018). The approach described in Oskiper, Zhu, Samarasekera, and Kumar (2007) uses two visual odometries, where only one is selected to be fused with an inertial measurement unit IMU. We fuse the pose estimates of each VO with the readings from the IMU in an extended, error-state Kalman filter.

2.4 | Motion planning

Motion planning can be categorized into different strategies, for example: global or local; path, kinematic or kinodynamic; and sampling-, graph- or optimization-based. For mobile robotic contexts, global planning strategies are typically path planning methods, which operate on a global map of the environment. We identified kinodynamic, global planning solutions as an important system criterion, since ARDEA's objective is exploration and ARDEA is subject to nontrivial system dynamics.

In unknown environments, random sampling based strategies, such as the well known rapidly exploring random tree (RRT; LaValle & Kuffner, 2001) or A* (Hart, Nilsson, & Raphael, 1968) algorithms can be modified to efficiently find collision-free paths in unstructured, dynamic environments. While they are advantageous for collision avoidance using discrete, noisy sensor data, optimization-based strategies can deliver optimal *and* feasible solutions w.r.t. a metric and system constraints. The 4 DOF flat representation of Mellinger and Kumar (2011) is widely used in the MAV community and is convenient for planning techniques, which can make use of locally optimized motion primitives. The minimal snap trajectories proposed by Mellinger and Kumar were extended in Richter, Bry, and Roy (2016) as the motion primitives for a sampling based search algorithm. Motion primitives in the flat system space were as well used as a fast primitive in a sampling based search method by M. W. Müller, Hehn, and D'Andrea (2015). Model predictive control (MPC) techniques were also used by Nägeli, Alonso-Mora, Domahidi, Rus, and Hilliges (2017) for online optimization of cinematic metrics over short time horizons.

Recently, there has been use of combinatorial strategies, which leverage the exploration capabilities of sampling and graph techniques to provide feasible initial guesses for optimal programming or direct trajectory optimization. Nieuwenhuisen and Behnke (2015) showed that they can efficiently plan control-effort optimal trajectories using as A*-based search in an Octomap (Hornung, Wurm, Bennewitz, Stachniss, & Burgard, 2013) and a subsequent smoothing using the CHOMP algorithm. More recently, Uszenko, vonStumberg, Pangercic, and Cremers (2017) and Oleynikova et al. (2016) showed that they can use on-board state estimation and mapping data from RGB-D and visual-inertial sensors, respectively, to build potential maps from Octomap occupancy trees and perform fast, online replanning using different optimization techniques.

2.5 | Control

Control of MAVs has been an active field of research. The control is important to execute the previously planned trajectories. A fundamental overview of multirotor control can be found in Mahony, Kumar, and Corke (2012). In Mellinger and Kumar (2011), aggressive flight maneuvers are realized based on differential flatness and using an external motion capture system. This approach is extended in Faessler, Franchi, and Scaramuzza (2018) to account for first-order drag effects. Another line of research focuses on nonlinear dynamic inversion, which uses the inverse of the dynamical model and generates a feedforward angular rate command based on a differentiable reference trajectory (M. W. Achtelik, Lynen, Chli, & Siegwart, 2013). The incremental version, incremental nonlinear dynamic inversion, uses stepwise updates of the control input and leads to commands and disturbance rejection on linear and angular acceleration level (Smeur, de Croon, & Chu, 2017). However, angular acceleration measurements are usually only available numerically on an MAV; hence, they are noisy and require filtering. Our

control strategy, in contrast to the aforementioned approaches, solely relies on VO. The classical cascade of position and attitude controller enables to manually fly the MAV in attitude control mode. We explicitly consider changes in the atmosphere and compensate for and distinguish between external contact and wind forces (Tomić, Lutz, Schmid, Mathers, & Haddadin, 2018).

3 | FUNDAMENTAL SYSTEM DESCRIPTION

In this section, we present the fundamental system setup of our MAV. First, in Section 3.1 the general hardware setup is described, including ARDEA's shape and electrical components. Here, we also discuss system design requirements with respect to navigation and exploration tasks for future planetary missions and present our design decisions. In Section 3.2 the various on-board sensors and their respective mechanical and electrical integration are described, and in Section 3.3 the propulsion system design is discussed. Finally, an overview of the low-level software is presented in Section 3.4, including operating systems, middleware and related components.

3.1 | General hardware and system software setup

In previous works, we used commercially available MAV platforms with modifications to make them suitable for our required objectives and sensor equipment (Schmid, Lutz, et al., 2014; Tomić et al., 2012). As mentioned in these publications, adaptation of a commercial platform is a good starting point for autonomous MAV research, but comes with several disadvantages. These include sensor arrangement limitations due to the fixed frame structure as well as limited access to low-level control interfaces of components such as the propulsion system.

3.1.1 | Requirements

To make this platform suitable for restricted spaces such as caves in planetary exploration scenarios or indoor environments such as partially collapsed man-made structures in SAR missions, the main design objective was to employ two pairs of wide FOV stereo cameras for visual-inertial navigation, without having propellers or frame components within the camera views. Besides the visual navigation aspect, the platform should also be suitable for control system research (Section 4.1.1), where access to low-level interfaces such as motor speed commands and telemetry are vital. Most commercial platforms do not provide such low-level interfaces, which renders them unsuitable for our control research. Therefore, it was required to design a custom system.

3.1.2 | Sensor placement considerations

Different system designs were taken into consideration in order for the wide-angle cameras to have an unobstructed FOV and to have the IMU in the center of gravity (CoG) of the mechanical structure. Moreover, mechanical decoupling of the sensors and propulsion system was a critical design criterion that motivated the separation of the system into two parts; a frame containing only the propulsion system and a navigation stack unit, as shown in Figure 2. All sensors, embedded computers and custom electronics are integrated into a stand-alone navigation stack as illustrated in Figure 3. Unlike the mechanical decoupling between frame and navigation stack, the IMU and stereo cameras must be rigidly mounted to each other to insure a high mechanical stiffness between them. This is critical because the translation and rotation between both sensors will be calibrated once and should stay constant during robot operation to ensure accurate visual-inertial navigation.

3.1.3 | Frame

The following shapes were considered as possible frame designs: Y, T, H, △, □, +, ×. Meaning that the frame resembles the shape of the letters and the rotors are situated at the extremities. Due to the requirement of mounting an exchangeable standalone navigation stack within the frame center, only shapes without edges passing through the center were considered, that is, △ and □. Comparing △ and □, the triangular arrangement gives the largest motor separation distance, and therefore, also the widest unobstructed camera FOV.

In favor of more stable propulsion, a coaxial motor arrangement with six motors in two planes was chosen, as shown in Figure 2a. This symmetric configuration of an equal number of counter-rotating motors on each plane takes care of the angular momentum balancing, which is common for multirotor system designs.

The propulsion system is mounted onto the frame and comprises electronic speed controllers (ESCs), motors, propellers and cabling. Landing gear and propeller guards (not shown on pictures) are as well mounted directly to the frame. The propeller guards are designed to shield not only the propellers but the exposed navigation cameras as well. For high stiffness, popular carbon fiber tubes with a diameter of 18 mm and a thickness of 0.55 mm were selected for the frame itself, whereas the landing gear and the propeller guards are attached with 8 mm diameter tubes. All power and data wires going from the navigation stack interface to the individual ESCs are contained in the tube structure. The carbon fiber tubes of the frame and landing gear are assembled with custom aluminum connection parts which also serve as motor mounts. After assembly the carbon fiber and aluminum parts are glued together with epoxy. The mechanical decoupling between the frame and the navigation stack is achieved by rubber dampers on the fixture for the navigation stack.

3.1.4 | Navigation stack

The navigation stack is a self-contained, detachable unit holding all sensors, embedded computers, and miscellaneous electronic components with the exception of the ESCs. It is a stand-alone unit in the

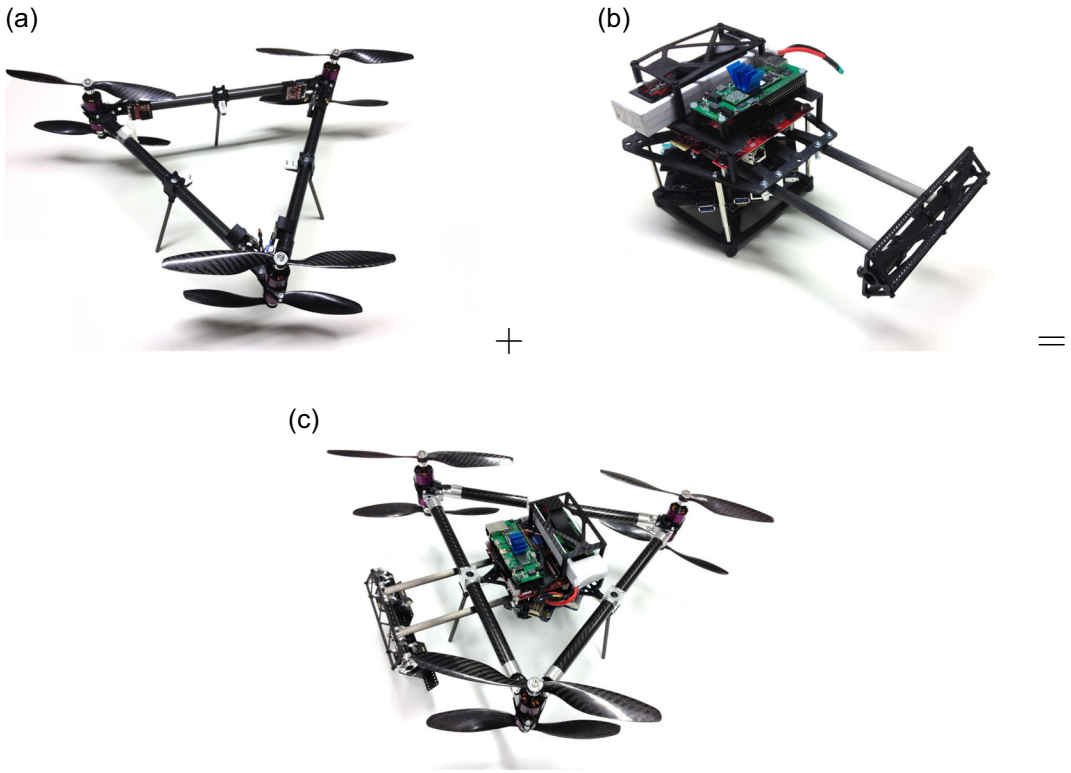


FIGURE 2 Modular flight stack implementation of an early prototype. (a) Open frame with landing gear and propulsion system; (b) navigation stack; (c) assembled frame and navigation stack [Color figure can be viewed at wileyonlinelibrary.com]

sense that it only needs supply voltage as input and provides a bi-directional controller area network (CAN-bus) data interface for controlling any actuators, for example, the ESCs.

Figure 2b depicts the navigation stack and Figure 3 illustrates the general hardware overview.

The navigation stack can also be used independently of the frame, either carried or attached to other types of mobile robots, to test navigation algorithms. It is comprised of the following components:

- Low-level real-time embedded computer: BeagleBone Black (BBB; singlecore 1 GHz CPU, 512 MB RAM) embedded single-board computer with a custom cape/breakout printed circuit board (PCB). It contains a watchdog safety circuit, power supplies for 3.3, 5, and 12 V and a buzzer.
- High-level embedded computer: Intel NUC5i7RYH (dual-core 3.1 GHz i7 CPU, 16 GB RAM).
- Spektrum 2.4 G Hz DSMX satellite RC receiver.
- Analog devices ADIS16367 IMU. It consists of 3 DOF accelerometers and 3 DOF gyroscopes, which are factory-calibrated and temperature compensated.
- Four wide-angle cameras for the dual stereo setup, see Section 3.2.
- Xilinx Spartan 6 LX75T FPGA running the SGM stereo algorithm by Hirschmüller (2008).
- Ubiquiti Bullet M5-HP 5 GHz WLAN access-point.

While all other components are rigidly mounted on the navigation stack, the IMU is mounted on rubber dampers in addition to the mechanical decoupling between frame and navigation stack. A dedicated low-level computer runs all hard real-time critical code such as the attitude and position controller along with the 500 Hz IMU readout. The high-level computer runs all computational intensive soft real-time critical visual navigation components such as VO, local reference filter and mission- and trajectory-planning.

The BBB custom breakout PCB contains the following components:

- Emergency power switch-off circuit, triggered by watchdog or explicit command. When turned on, the n-channel metal-oxide-semiconductor field-effect transistor (MOSFET) circuit has a 0.5 mΩ low impedance resistance for low heat dissipation.
- CAN-bus driver circuitry with switch-off from ESC data signals.
- Soft start with smooth current ramp up while power is switched on, to limit maximum current drawn by ESCs, which acts as high capacitive load.
- Trigger functionality for driving external illumination LEDs to aid vision in poorly lit conditions.
- Interface break-out for ESCs, IMU, camera-trigger, buzzer, status display, and LEDs.

3.2 | Vision sensor setup

The stereo camera pairs are the primary sensors on-board ARDEA and their output is used for flight critical as well as higher-level

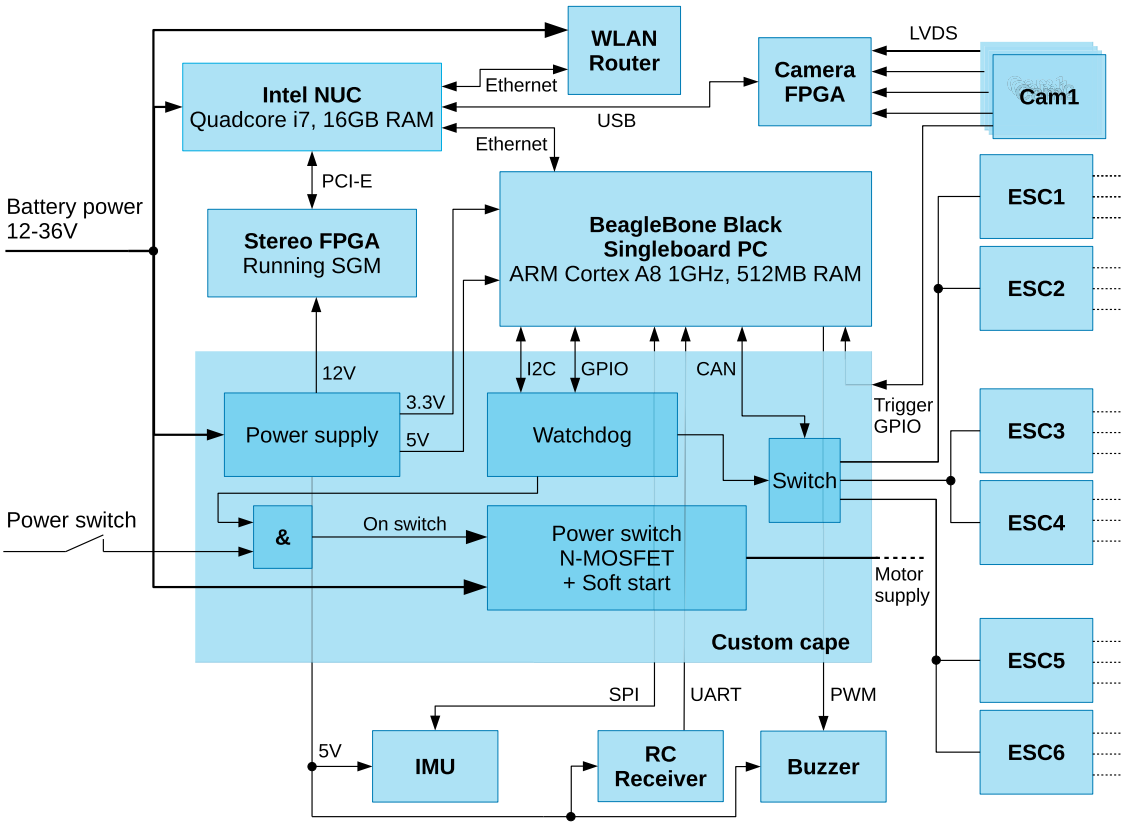


FIGURE 3 General hardware interconnection overview of autonomous robot design for extraterrestrial applications [Color figure can be viewed at wileyonlinelibrary.com]

autonomy software, including state estimation and map generation. A central application for the MAV is exploration of natural caves. To efficiently explore and map unknown environments such as caves while ensuring collision-free motion, it is necessary to perceive as much as possible. Cameras with wide-angle lenses arranged in a stereo setup are well suited for this purpose. To achieve a large vertical FOV with reasonable resolution we use two wide-angle camera pairs aligned such that the larger image dimension lies along ARDEA's vertical body axis.

We chose VRmagic cameras containing VRmMFC camera base units and four VRmS-16/C-COB sensor boards with wide angle lenses, each providing a FOV of approximately 80° horizontally and 125° vertically. The cameras are arranged as two stereo systems with the optical axes of the lower camera system at a −60° angle with respect to the horizon and those of the upper cameras at +60° (see Figure 4). As a result the complete stereo setup provides approximately 240° vertical field of view as illustrated in Figure 5.

In addition to the advantages this camera setup has in indoor scenarios, the arrangement of cameras is also well suited for the high dynamic range situation in outdoor scenes with often much higher brightness above the horizon than below. As separate cameras cover the FOV below and above the horizon, longer exposure times and higher gains can be used for the lower FOV to cope with the different intensities. The camera base unit triggers the cameras synchronously, captures all images, applies preprocessing, and sends them to the NUC high-level embedded computer via USB. This hardware trigger is also connected to the BBB computer, which saves the current timestamp along with the last IMU values for every trigger event and sends it via message to the local navigation filter.

As illustrated in Figure 6 and described in detail in M. G. Müller et al. (2018), each image is remapped to two separate images with pinhole projections, resulting in four images from left cameras and four images from right cameras. Images from both sides are then stacked together to one left and right image and processed on an FPGA to obtain depth information by means of the SGM stereo algorithm (see Figure 7).

3.3 | Propulsion system

Although the here presented conceptual robot is designed for future planetary missions, its propulsion system is layout to operate in Earth atmosphere to test the autonomy software and overall system. As previously introduced in Section 3.1.3, we chose a symmetrical, coaxial tricopter layout with two times three motors in a coplanar arrangement to balance the resultant angular momentum of the

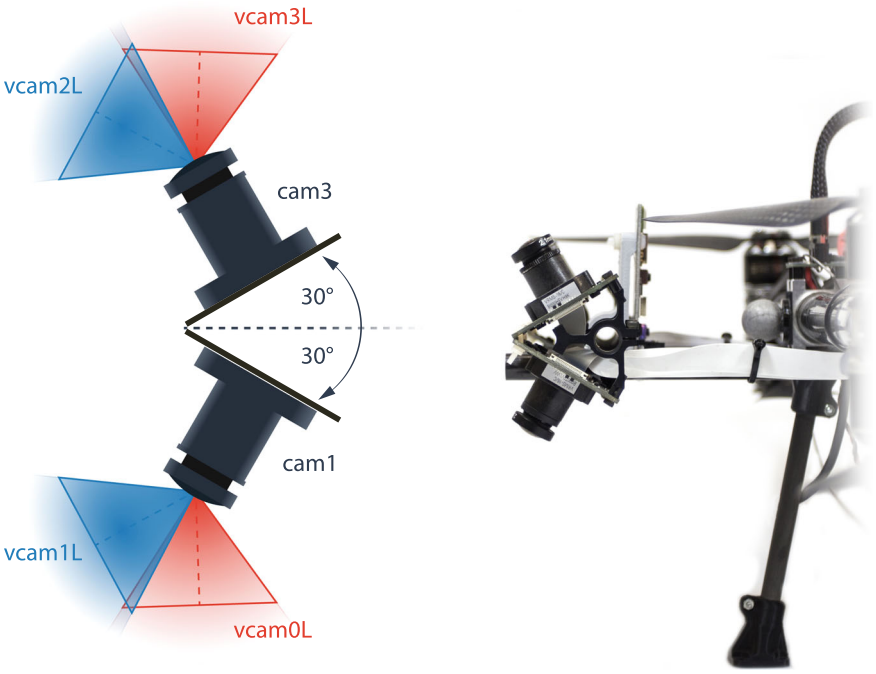


FIGURE 4 Left: Lateral view of the left side of the multicamera system. From each physical wide-angle camera (*cam1*, *cam3*), two "virtual" pinhole cameras are synthesized by rotating ±30° around the stereo axes (*vcam0L* and *vcam1L* belong to *cam1*, *vcam2L*, and *vcam3L* belong to *cam3*). On the mirror symmetric right side, the physical cameras *cam2* and *cam4* are located and remapped to *vcam0R*/*vcam1R* and *vcam2R*/*vcam3R*, respectively. Each virtual pinhole camera has a FOV of 80° × 65°; the total vertical FOV of the stereo camera system is approx. 240°. Right: Photo of the real camera setup [Color figure can be viewed at wileyonlinelibrary.com]

system in an implicit fashion. This common design practice allows multirotor systems to achieve stable propulsion. In the following sections we outline a rule-of-thumb description of the system design criterions for both the aerodynamic as well as the electrical communication design aspects to enable bidirectional communication with the least required amount of cables to keep the system simple and lightweight. The actual implementations of the attitude and position controllers are discussed in detail in Section 4.1.1.

3.3.1 | Aerodynamic design considerations

By applying the thrust equation in Equation (1) we estimated the approximate propeller diameter D and rotor speeds ϖ_i of rotor $i \in [1, ..., 6]$ which would be necessary to achieve hover conditions.

$$T = \sum_{i=1}^{6} \rho C_{T,i} D^4 \varpi_i^2. \tag{1}$$

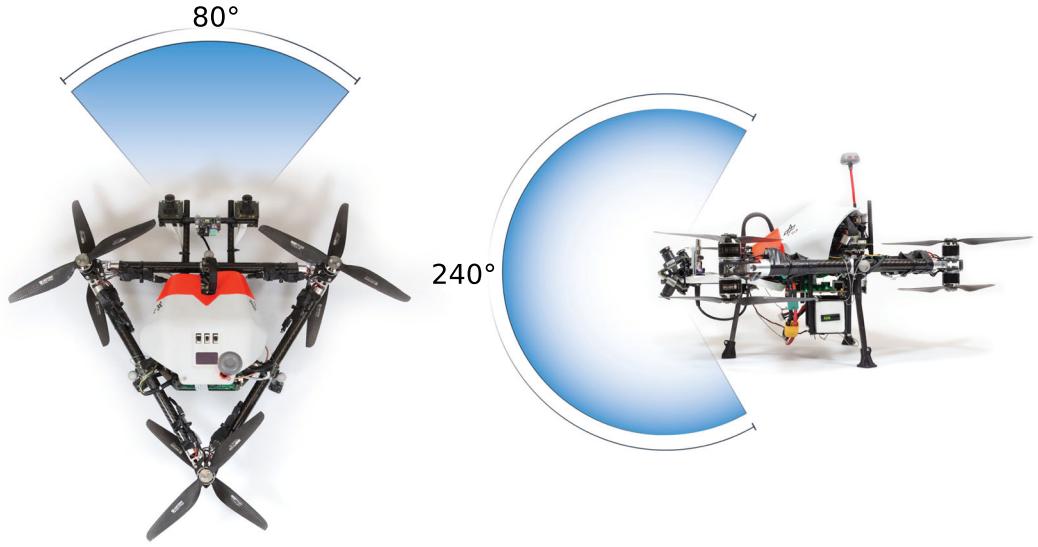


FIGURE 5 Top and side view of ARDEA. Bluish areas indicate the horizontal and vertical field of view of the multicamera system [Color figure can be viewed at wileyonlinelibrary.com]

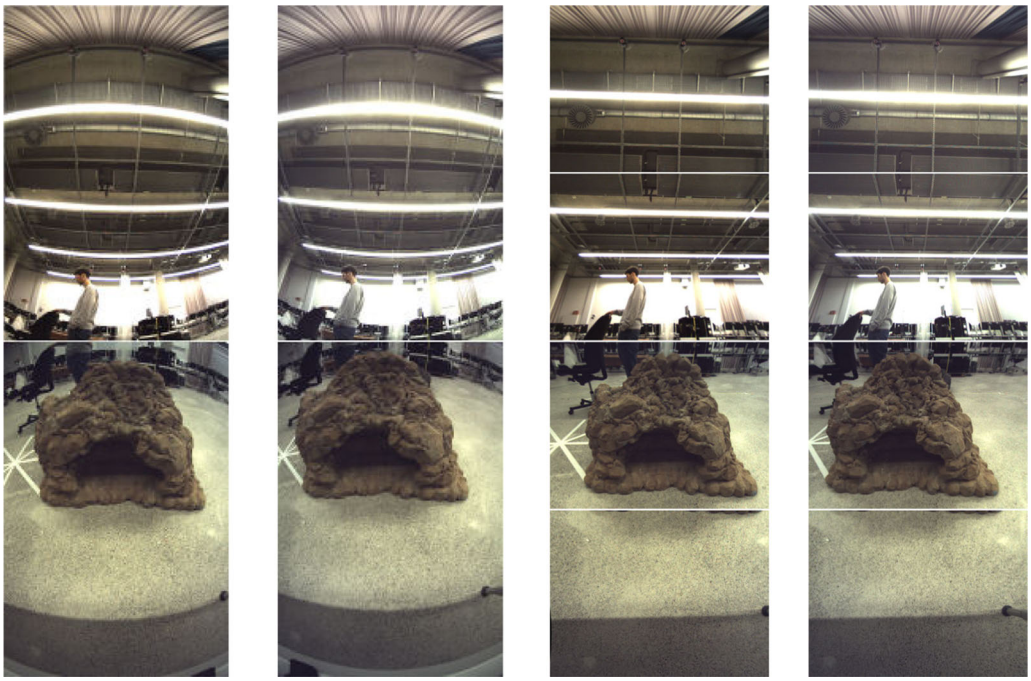


FIGURE 6 Raw and pinhole camera images. From left to right: raw images of camera 1 (bottom) and camera 3 (top); raw images of camera 2 (bottom) and camera 4 (top); remapped pinhole images of virtual cameras 0L,1L,2L,3L (from bottom to top); remapped pinhole images of virtual cameras 0R,1R,2R,3R. Note that "straight lines are straight" after remapping to pinhole images, but appear bent in the original camera images [Color figure can be viewed at wileyonlinelibrary.com]

The nondimensional thrust coefficient C_T must be identified for the exact hardware configuration, which we discuss later in Section 4.1.1. Due to spatial constraints originating from the navigation stack design (see Section 3.1.4), the maximum propeller diameter is limited to 28 cm. For that reason a 10-inch propeller (T-Motor 10 × 3.3 inch CF) was chosen with low inertia, due to the carbon fiber material, and a low propeller slope of 3.3, because of better efficiency in hover and at low forward velocities. The platform as a whole was specified to weigh roughly 2.6 kg, resulting in a simplified propulsion model of approximately 4.16 N thrust per motor. Rearranging Equation (1) to calculate the minimum required ϖ_i for hover conditions and using $\rho = 1.2\ \mathrm{kg\ m^{-3}}$ (dry air at 20°C), $C_T = 0.01$ and $D = 0.254$ m yields approximately 6,000 rpm per motor. As a rule of thumb, we account for an additional 50% of that thrust for maneuverability and for losses introduced by the coaxial rotor configuration, resulting in a minimum required propeller angular speed of 9,000 rpm. The selection of the motors was driven by the motor velocity constant K_v, which is measured in $\mathrm{rpm\ V^{-1}}$ for a motor without load, that is, without a propeller. Using Li-Po batteries with four cells and assuming a cell voltage of 3.6 V for an empty and 4.15 V for a full battery results in an usable voltage range of 14.4–16.8 V. The lower voltage value defines the minimum required

FIGURE 7 Field programmable gate array stereo processing: left input image consisting of the pinhole images of all left virtual cameras, that is, 0L,1L,2L,3L, arranged in a 2 × 2 grid. The resulting depth map is shown on the right [Color figure can be viewed at wileyonlinelibrary.com]

K_v motor constant of at least 625 rpm V^{-1}. Having a significantly higher K_v for the loaded motor case with propeller attached made us choose a T-Motor MN2212 with a K_v value of 920 rpm V^{-1}.

3.3.2 | Electrical and communication design

As previously mentioned our supply voltage is provided by a 4-cell Li-Po accumulator. To incorporate the rotor angular speeds into the feedback loop of the attitude controller an ESC which is providing those values over a telemetry link is required. There are two common ESCs interfaces:

1. Point-to-point serial interface, for example, RS232, 1-wire.
2. Full- or half-duplex data bus realizations such as RS485 or CAN-bus.

Moreover, the latter communication approach replaces the common PWM servo position or the newer oneshot, multishot, or DShot protocols which are used to send speed commands to ESCs. Motor controllers like modern KISS ESCs provide a telemetry link via a 1-wire bus, that means for n ESCs one needs n wires for sending commands, one for receiving telemetry over 1-wire and one for the signal reference (GND), resulting in eight wires for the hexacopter platform design which is discussed here. This communication scheme is not optimal in the way it uses an excess of cables, which adds design complexity and additional weight to the system. By comparison, bidirectional data buses like RS485 or CAN-bus are based on a physical layer using differential signals, and therefore, do not need a common reference voltage (GND). They can be implemented in a half-duplex fashion, requiring only two cables for sending motor commands and receiving telemetry over the same wires. Unlike RS485, the CAN-bus standard not only specifies the physical layer but also the data link layer (according to the ISO/OSI model), which has many favorable properties, such as:

- Multimaster bus: Reduces need for time-consuming polling in a master-slave system. Telemetry messages are sent on the bus without having cyclic request messages.
- Multicast reception with time synchronization: Synchronous sending of motor commands to all ESCs in one message reduces overhead.
- Error detection and signaling: A faulty bus state can be detected and recovered automatically.
- Bus access collision avoidance through prioritization of messages: Simplifies communication protocol design by not having to take care of it.

Those properties simplify the communication design complexity and therefore, give CAN-bus a clear advantage for our desired platform design. Galvanic isolation to alleviate the maximum common mode voltage range limit of CAN-bus transceivers is not necessary because the range of our selected CAN-bus transceiver (−7 to +12 V) proved to be sufficient for stable communication even in noisy conditions caused by fast switching of currents up to 10 A within the ESCs.

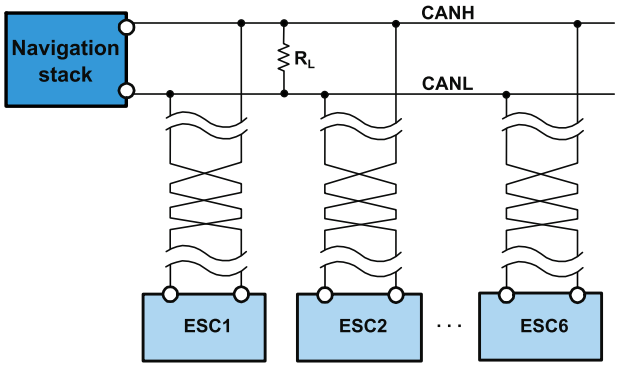


FIGURE 8 CAN-bus topology showing the multiple termination concept on all stub lines [Color figure can be viewed at wileyonlinelibrary.com]

Although star topologies are not recommended due to causing signal reflections, they offer a more flexible way of routing the data bus cables across the platform frame and they are unproblematic, if the cable lengths are short enough. Our design is according to the high-speed ISO 11898 standard specifications, which recommend a maximum unterminated stub length of 0.3 m with a 1 Mbit/s data rate. Figure 8 shows the bus topology and the bus termination strategy with only one resistor (R_L = 120 Ω) on the BBB custom PCB (see Section 3.1.4). This is not according to the standard but neglecting reflections due to stub lines in a star topology is a convenient design simplification for the short wires in our design. Only one termination resistor is used to match the bus on the PCB to the twisted-pair cable impedance and to attenuate present reflections.

At the time of designing the platform, the only commercially available ESC which provided CAN-bus communication was the ESC32v3, while at the time of writing this manuscript there are two more choices: Zubax "Myxa" and "Orel20" ESCs[2] and an open-hardware and open-source ESC called VESC[3]. In the meantime the community even established a dedicated lightweight CAN-bus protocol layer intended for interfacing UAV components such as ESCs, sensors and other actuators, called UAVCAN[4].

3.4 | System software setup

To enable true autonomy our MAV has to run all processing on-board, this includes sensor processing, mapping and planning. Similarly to Schmid, Lutz, et al. (2014) we are separating autonomy functionality into low-level, real-time (RT) and high-level, computationally intensive and high latency tasks without hard real-time constraints, which is discussed in more detail in Section 4. RT critical software modules are the attitude- and position-controller, and the respective sensor driver modules responsible for IMU readout.

[2]Zubax ESCs: https://zubax.com/products

[3]Vedder ESC: http://vedder.se

[4]UAVCAN website: https://uavcan.org

To have the same operating system (OS) application programming interface (API) for all autonomy software components (see Section 4) we employ Linux in two flavors:

- RT: Ubuntu Linux 14.04 PREEMPT_RT Linux Kernel, running on the low-level ARM-based BBB computer.
- Non-RT: openSUSE Leap 42.3 with vanilla Linux Kernel, running on the high-level x86 NUC computer.

The latter OS is also used on all developer workstation x86 PCs, sharing the same CPU architecture and therefore, guaranteeing binary compatibility. This enables fast software development cycles, software packages can be compiled on workstation PCs and directly deployed on the robot, without needing a cross-compilation workflow.

Along with the RT OS, also the middleware has to be capable of providing deterministic timing for RT communication, for example, in the time sensitive control feedback loop. Therefore, we employ a custom, in-house RT middleware allowing publish-subscribe and service call communication schemes. For high-level and less time critical software components (see Section 4.2) we use the robot operating system (ROS) middleware.

Because sensor data is acquired on both the BBB and NUC computer, their system clocks have to be time synchronized for consistent data association in the local reference filter, which is done in the local navigation filter (see Section 4.2.2). The precision time protocol daemon (PTPd) is used for fast and accurate system clock synchronization, it is started with a rate of 4 Hz, allowing time stepping upon startup and fast clock slewing during system initialization and achieves a clock synchronization accuracy of below 1 ms within half a minute. Moreover, camera trigger timestamps are captured on the BBB computer (see Figure 3) for accurate state vector augmentations in the local navigation filter.

4 | AUTONOMY SOFTWARE

Our autonomy software is separated into low-level and high-level components. The former is comprised of elementary functionality such as system stabilization in the attitude and position controller for manual piloting. Those algorithms have to run on a real-time OS with a high priority to satisfy the tight task scheduling requirements of the control loop, running at 500 Hz. Because large deviations of the task scheduling latency lead to system instability these software parts can not run on a remote PC and have to run on-board the system.

The latter software type deals with higher level tasks such as navigation and mapping. These algorithms are computationally intensive and hence characterized by high latency. Due to their high latency (typically 50–200 ms) they are not real-time critical and could even run on separate ground-station PCs, while WLAN communication latencies (typically 1–20 ms) can be neglected. In a robotic team those computations could as well be transferred to team members which have higher computational power. The higher level autonomy software adds autonomy on top of the lower level one, which makes the MAV operational for autonomous missions. In our case, the higher level tasks are executed on-board the MAV, since we cannot rely on any communication with a ground robot or station in a planetary mission setup. All low-level algorithms are running on the BBB real-time computer, whereas the high-level part is running on the Intel NUC and FPGAs.

Figure 9 gives an overview of our software architecture and how autonomy software components are grouped into *perception*, *localization and mapping*, and *planning and control* layers. Please note that not all details are captured by this high-level overview, for a more detailed insight into the mentioned blocks refer to the respective sections. The flow of data starts in the *sensor* layer at the top and goes all the way down to the *actuators* layer, whereas dashed boxes symbolize hardware with their respective low-level software interface. Acquisition and processing of camera images is done within the *perception* layer, essentially extracting a sparser representation from the information rich camera streams for the *localization and mapping* components. Within the *localization and mapping* layer this preprocessed sensor data is used for local and global navigation and mapping. Lastly, the *planning and control* layer is involved in stabilizing, planning, and moving the MAV in the desired manner. The high-level skills are realized within our RMC advanced flow control (RAFCON) state machine framework and are connected across all three high-level software layers.

4.1 | Low-level autonomy software

In this section, we describe the low level autonomy software on ARDEA. All of the here described functionalities are running on-board the MAV with real-time processing constraints. First the basic control system is described, following up with the external wrench estimation, which makes the vehicle observe forces and torques caused by contacts and wind influence for improving robustness. The latter is important for missions performed on planetary bodies with atmosphere.

4.1.1 | Controller

As any multicopter with parallel thrust vectors, our ARDEA hexacopter is underactuated, that is, it has four control inputs (collective thrust and three torques) but six DOF. However, the system is differentially flat, meaning that the control input can be computed from the so-called flat outputs (position $\boldsymbol{p}$ and heading ψ) and their derivatives (Faessler et al., 2018; Mellinger & Kumar, 2011). Hence, to control position and heading of ARDEA, we utilize a cascaded structure of position and attitude controller as depicted in Figures 10, and 11. It has the advantage that the attitude controller can be used even if a position controller is not present, for example, if the MAV is manually piloted. The control software was implemented using an in-house microthread management C++ library which takes care of individual scheduling of software blocks, the interface, and data flow between them.

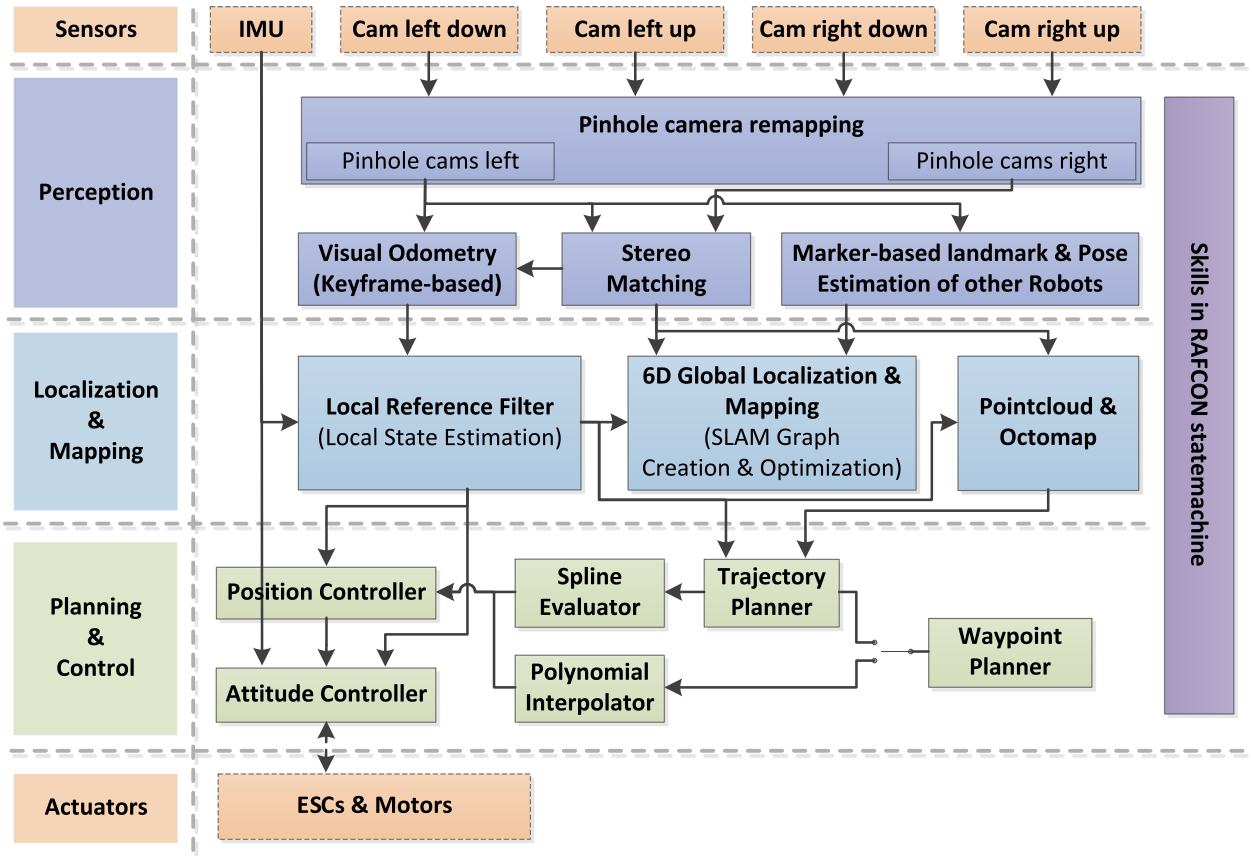


FIGURE 9 Autonomy software architecture overview. Boxes with dashed lines symbolize hardware with their respective low-level software interface [Color figure can be viewed at wileyonlinelibrary.com]

Dynamics model

The rigid-body model of ARDEA used for control is given by well-known Newton Euler equations (Tomić, Ott, & Haddadin, 2017) as

$$m\ddot{\boldsymbol{p}} = mg\boldsymbol{e}_3 + T\boldsymbol{R}\boldsymbol{e}_3 + \boldsymbol{R}\boldsymbol{f}_e, \tag{2}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = \boldsymbol{S}(\boldsymbol{J}\boldsymbol{\omega})\boldsymbol{\omega} + \tau_{cog} + \tau + \tau_e, \tag{3}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega}), \tag{4}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position in the inertial frame, $\boldsymbol{\omega}$ is the angular velocity of the body w.r.t. the inertial frame, $\boldsymbol{R}$ is a rotation matrix, $\boldsymbol{e}_3 = (0 \quad 0 \quad 1)^T$ is a unit vector, $\boldsymbol{S}(\cdot)$ is the matrix representation of the cross-product, and $T \in \mathbb{R}$ and $\tau \in \mathbb{R}^3$ are the control inputs thrust and body-torque about the geometric center of propellers (frame B), respectively.

The CoG offset from the geometric center of ARDEA is considered by the term $\tau_{cog} = mg\boldsymbol{S}(\boldsymbol{r}_g)\boldsymbol{R}^T\boldsymbol{e}_3$. The external force $\boldsymbol{f}_e$ and external torque τ_e form the external wrench $\boldsymbol{w}_e = [\boldsymbol{f}_e^T \ \tau_e^T]^T$. This lumped term $\boldsymbol{w}_e = \boldsymbol{w}_m + \boldsymbol{w}_d(\boldsymbol{v}_r) + \boldsymbol{w}_c$ captures modeling errors $\boldsymbol{w}_m$ (including faults), external disturbances $\boldsymbol{w}_d(\boldsymbol{v}_r)$ resulting from aerodynamics and dependent on the relative airspeed $\boldsymbol{v}_r = \dot{\boldsymbol{v}} - \boldsymbol{v}_w$, with $\boldsymbol{v}_w$ being the wind velocity. The term $\boldsymbol{w}_d$ thus depends on the aerodynamics model and surrounding airflow. Finally, $\boldsymbol{w}_e$ also captures the physical interaction wrench $\boldsymbol{w}_c$. The vehicle controller has to compensate these external influences to track a reference trajectory. However, the external

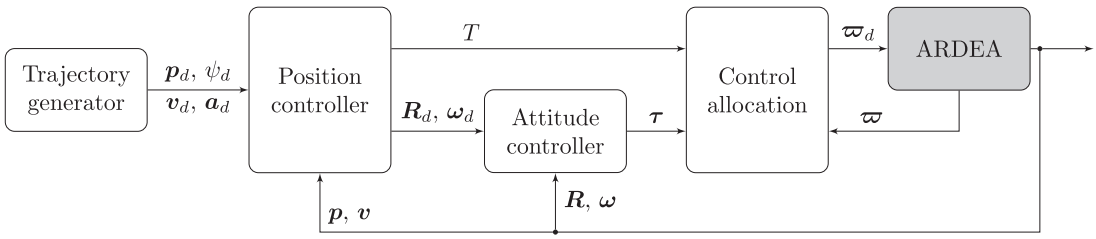


FIGURE 10 Structure of the cascaded position and attitude controller

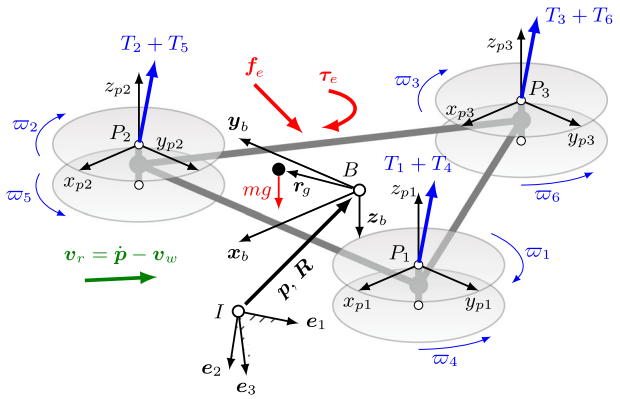


FIGURE 11 Schematic depiction of the autonomous robot design for extraterrestrial application model and configuration [Color figure can be viewed at wileyonlinelibrary.com]

wrench can be estimated (Section 4.1.2) and its components discriminated. The accuracy of the wrench estimate may be increased using the adaptive air density estimation skill, see Section 5.4. From the obtained wrench and rotor power measurements, the surrounding wind velocity may also be estimated. The identification process of the remaining static model parameters is explained in the next paragraph and the resulting values are summarized in Table 1. In the following, the attitude and position controller are presented under the assumption of no external disturbances, that is, $\boldsymbol{f}_e = \boldsymbol{0}, \tau_e = \boldsymbol{0}$.

Control allocation

The control allocation maps the computed thrust T from the position controller as well as the computed torques τ from the attitude controller to rotor speeds ϖ_i of the six propellers, $i \in [1, ..., 6]$, such that

$$\begin{pmatrix} \varpi_1^2 \\ \vdots \\ \varpi_6^2 \end{pmatrix} = \frac{1}{\rho} \boldsymbol{W} \boldsymbol{B}^{\#} \boldsymbol{\Sigma} \begin{pmatrix} T \\ \boldsymbol{\tau} \end{pmatrix}, \tag{5}$$

TABLE 1 ARDEA model parameters

Parameter		Value
Mass	m	2.68 kg
Circumcircle radius of frame	r	0.23 m
Propeller diameter	D	0.254 m
Lower motor torque coefficients	$K_{q,0}$	[2.9404, 1.4545] $\cdot 10^{-2} N\ m\ A^{-1}$
Upper motor torque coefficients	$K_{q,1}$	[−1.4099, −3.3360] $\cdot 10^{-3} N\ m\ A^{-2}$
Thrust coefficient	C_T	[0.046457, 0.071021]
Torque coefficient	C_Q	[0.0075183, 0.0047597]

Note: Coaxial propeller pair coefficients are written as [upper, lower]. The inertia and center of gravity position were identified using data from an identification flight from Tomić et al. (2017) and identified propulsion parameters.

where ρ is the air density, $\boldsymbol{W}$ is a diagonal matrix, where the elements w_{ii} are either 1 or 0 on motor failure, $\boldsymbol{B}^{\#}$ is a generalized inverse of $\boldsymbol{B}$, and $\boldsymbol{\Sigma}$ is a diagonal scaling matrix used for saturation handling. The elements Σ_{ii} are computed iteratively if maximum motor speeds are reached. The allocation matrix $\boldsymbol{B}$ depends on rotor thrust and torque coefficients and on the circumcircle radius r of the triangular frame

$$\boldsymbol{B} = \begin{bmatrix} -c_u & -c_u & -c_u & -c_l & -c_l & -c_l \\ \frac{\sqrt{3}\ c_u\ r}{2} & -\frac{\sqrt{3}\ c_u\ r}{2} & 0 & \frac{\sqrt{3}\ c_l\ r}{2} & -\frac{\sqrt{3}\ c_l\ r}{2} & 0 \\ \frac{c_u\ r}{2} & \frac{c_u\ r}{2} & -c_u\ r & \frac{c_l\ r}{2} & \frac{c_l\ r}{2} & -c_l\ r \\ -k_u & -k_u & -k_u & k_l & k_l & k_l \end{bmatrix}. \tag{6}$$

Note that for convenience, the parameters rotor radius and rotor disk area may be lumped in the thrust and torque coefficients c_u, c_l, k_u, k_l of upper and lower propeller, respectively. The coefficients of the propellers used on ARDEA were identified based on force-torque sensor experiments (Tomić, Schmid, Lutz, Mathers, & Haddadin, 2016). For the quadratic model Equation (5), it was found that upper and lower propellers in the coaxial configuration have different coefficients, which is due to the fact that the lower propeller operates in the downstream of the upper propeller. Our control allocation is implemented generically for different multirotor configurations, for example, quadrocopters, hexacopters, or octacopters. Hence, although an analytical solution $\boldsymbol{B}^{\#}$ exists, we compute it numerically once during startup and as soon as reallocation is required, for example, if a motor fails or a propeller is lost. A single rotor is removed from the control allocation by deleting the respective column in $\boldsymbol{B}$ and the numerical pseudoinverse is obtained using singular value decomposition (SVD). Detection of motor failures on ARDEA is possible because of motor telemetry transmitted via CAN-bus (see Section 3.3.2). This increases fail-safe robustness (Michieletto et al., 2018) in the sense that ARDEA can maintain stable flight with less than six propellers, provided that the cumulative thrust of the remaining motors is sufficient for the actual takeoff weight. However, an equilibrium of rotor forces and moments is only obtained for a set of four remaining rotors that satisfies $\boldsymbol{B}\boldsymbol{B}^{\#} = \boldsymbol{I}$. Otherwise the vehicle will end up rotating at a defined rate (M. Müller & D'Andrea, 2014).

Parameter identification

Identification of the rigid body parameters is based on the linear parameterization

$$\boldsymbol{Y}\theta = \boldsymbol{u}, \tag{7}$$

where $\boldsymbol{Y} \in \mathbb{R}^{N \times M}$ is the regression matrix, $\theta \in \mathbb{R}^M$ is the vector of unknown parameters, and $\boldsymbol{u} \in \mathbb{R}^N$ is the known input. Therein, N is the number of measurement samples, and M is the number of parameters. For the dynamics model of ARDEA the following parameters have to be identified: mass m (one parameter), inertia $\boldsymbol{J}$, where we only consider the diagonal elements (three parameters), center of gravity $\boldsymbol{r}_g$ (three parameters), thrust and torque coefficients

of the coaxial propellers (four parameters), propeller inertia (one parameter), and the motor torque coefficients (two parameters), resulting in a total of 15 parameters. The parameter estimation is performed in three steps as shown in Figure 13, to reduce the search space. Parameters are identified by minimizing the ℓ_1 norm of the model residuals using the iteratively reweighted least squares (IRLS) method (Chartrand & Yin, 2008), which is robust against outliers.

Identifying all parameters from flight data is difficult due to the lack of ground truth measurements of the total torque acting on the vehicle. In addition to that, we found that identifying the thrust and torque coefficients from flight data is sensitive to time delay in the measurements (on the order of 20 ms), and can lead to physically meaningless parameters, like negative thrust coefficients. Hence, we split the identification into three parts as depicted in Figure 13. The propulsion model serves as ground truth for the rigid body identification. In the last step, the external wrench is identified based on the previously identified parameters. During our experiments, only the aerodynamic wrench acts on the robot, hence, we use the estimated external wrench to identify the aerodynamic model.

Propulsion system parameters

The propulsion parameters of ARDEA are stacked in the vector

$$\theta_1 := \left[C_{T,u},\ C_{T,l},\ C_{Q,u},\ C_{Q,l},\ \boldsymbol{K}_{q,u}^T,\ \boldsymbol{K}_{q,l}^T,\ I_r\right]^T, \tag{8}$$

and the regression matrix $\boldsymbol{Y}_1$ contains the rotor rates and motor current. The motor torque coefficients $\boldsymbol{K}_{q,i} = [K_{q,0i}, K_{q,1i}]^T$ are split for upper and lower motors ($\boldsymbol{K}_{q,u}$ and $\boldsymbol{K}_{q,l}$, respectively) because of the aerodynamic interaction between the propellers in the coaxial configuration. For this step, we fixed the hexacopter to an ATI 85 Mini force-torque sensor as depicted in Figures 12 and 13. The wrench measured by the sensor is denoted as $\boldsymbol{u}_1 := \tau_{\text{FTS}}$. We logged the pose provided by a motion capture system at 250 Hz, IMU data, motor speed, and current as measured by the speed controllers, the commanded control input, and the measured force and torque. The onboard attitude controller ran at 500 Hz. We calibrated the relative orientation of the force-torque sensor to the IMU beforehand. The resulting parameter estimates are listed in Table 1. Figure 14 shows a comparison of the identified model to the force-torque sensor measurements. It can be seen that the identified propulsion model closely matches the force-torque sensor measurements. Here, using the measured motor speeds to obtain the control wrench only results in a minor improvement compared to using the commanded speeds.

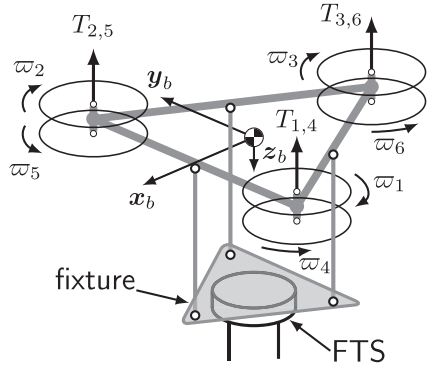


FIGURE 12 FTS setup for identification of propulsion parameters (Tomić et al., 2017, 2018)

For the yaw torque, we consider the estimated rotor acceleration $\dot{\varpi}$ to account for fast transitions and, therefore, improve the accuracy of the estimate (cf. Figure 15a). Note that adding the measured motor current does not increase the accuracy over using only the rotor acceleration. However, in the case of actuator failure (e.g., partially losing a propeller), the motor current will provide a better estimate of the yaw torque, as the method does not explicitly consider the propeller drag torque. Figure 15a depicts a comparison of the estimation using different measurements as well as the motor current.

Rigid body parameters

Considering a diagonal inertia tensor, the vector of rigid body parameters is given by

$$\theta_2 := \left[J_{xx},\ J_{yy},\ J_{zz},\ \boldsymbol{r}_g^T\right]^T. \tag{9}$$

The input $\boldsymbol{u}_2$ is obtained from the identified propulsion model and the known mass m such that

$$\boldsymbol{u}_2 = \boldsymbol{Y}_1\theta_1 - m\boldsymbol{y}_m, \tag{10}$$

where $\boldsymbol{y}_m$ is the regression matrix column associated with the mass. Table 2 lists the identified parameters and the propulsion model torque is compared to the torque predicted by the identified rigid body model in Figure 14. We find that the ℓ_1-identified parameters match closely to the least squares ℓ_2 parameters, which confirms the correctness of the identified dynamics model. Identification of the aerodynamic model was done through wind tunnel experiments, this is beyond the scope of this paper but described in more detail in Tomić et al. (2018).

Attitude controller

For attitude control, we employ a model-based proportional derivative (PD) controller similar to the one presented in Tomić et al. (2017), which utilizes the geometric attitude error $\boldsymbol{e}_R = \left(\boldsymbol{R}_d^T\boldsymbol{R} - \boldsymbol{R}^T\boldsymbol{R}_d\right)^\vee$ between the measured orientation $\boldsymbol{R}$ and the desired orientation $\boldsymbol{R}_d$ and with $(\cdot)^\vee$ being the vee map (Lee, Leoky, & McClamroch, 2010)

$$\tau = \boldsymbol{J}\left(-\boldsymbol{K}_\omega(\omega - \omega_d) - \frac{1}{2}\boldsymbol{C}\boldsymbol{e}_R\right) - \boldsymbol{S}(\boldsymbol{J}\omega)\omega. \tag{11}$$

Position controller

The position controller is also a model-based PD controller as in Tomić et al. (2017)

$$\boldsymbol{f}_i = m(\ddot{\boldsymbol{p}}_d + \boldsymbol{K}_d(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \boldsymbol{K}_p(\boldsymbol{p}_d - \boldsymbol{p}) - g\boldsymbol{e}_3). \tag{12}$$

It feed-forwards the desired linear acceleration $\ddot{\boldsymbol{p}}_d$ of the reference trajectory, which is generated either by the naive polynomial

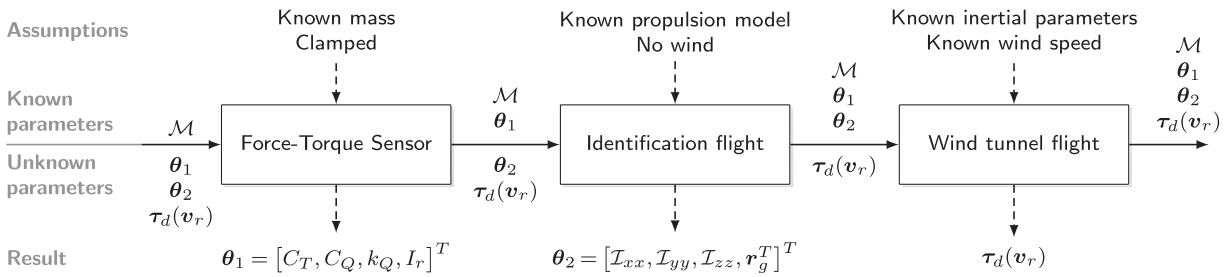


FIGURE 13 Parameter identification procedure according to Tomić et al. (2018). The procedure is done in three steps to minimize coupling effects in the high-dimensional parameter space

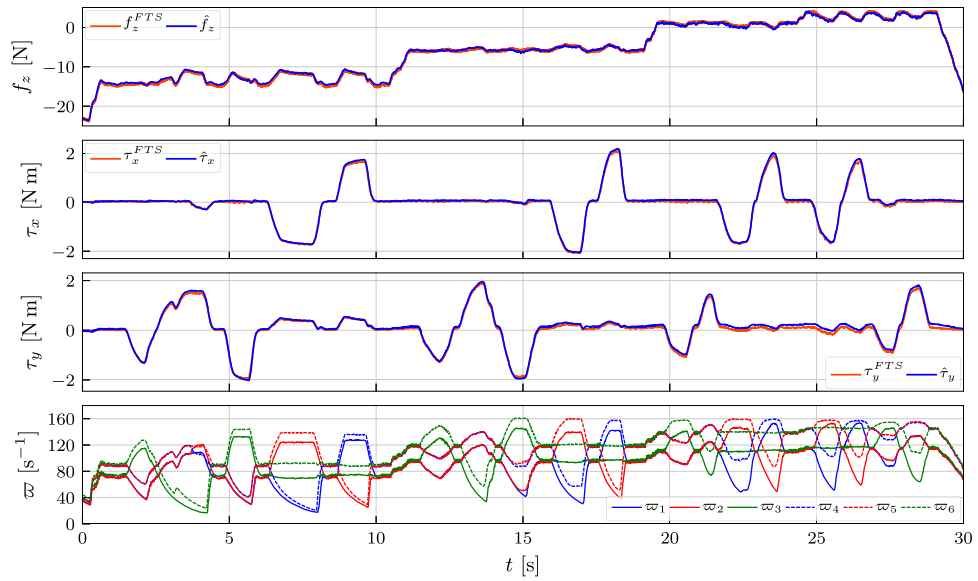


FIGURE 14 Validation of the propulsion model on the force-torque sensor setup depicted in Figure 12 after obtaining C_T and C_Q. The propulsion forces and torques are obtained using the measured motor speeds, shown in the bottom plot. The measured motor speeds of the upper propellers are shown as solid lines and lower propellers are shown as dashed lines [Color figure can be viewed at wileyonlinelibrary.com]

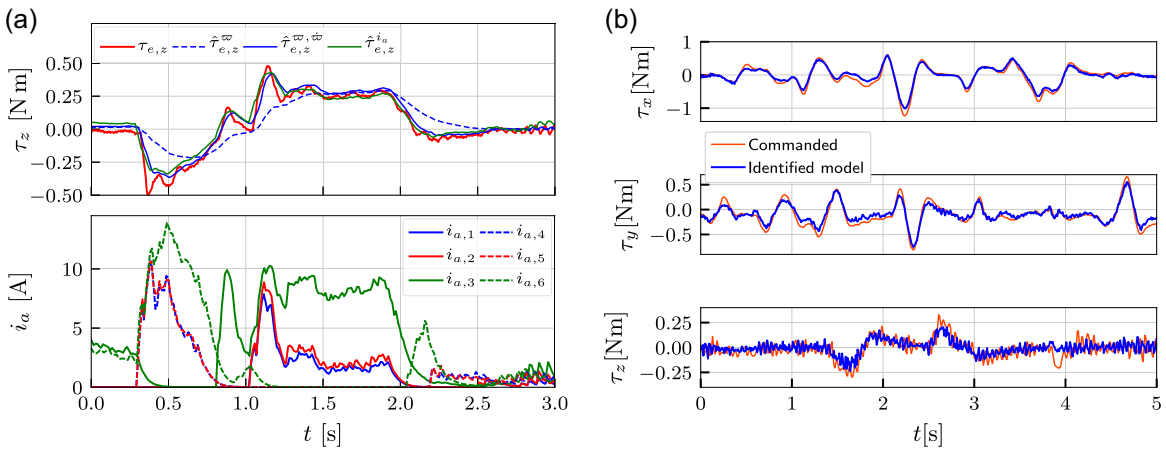


FIGURE 15 Validation of the dynamics model. The yaw torque can be estimated using rotor acceleration $\dot{\varpi}$ and the identified rotor inertia J_r, as well as the current i_a as measured by the ESC, and the identified motor torque coefficient k_T. Rigid body parameters are identified using data from an indoor flight at low airspeeds but high accelerations to excite the dynamics parameters. Ideally, the rigid body forces and torques should match the now identified and known propulsion inputs, which follow the near-hover model due to low airspeeds. The right plot compares the commanded propeller torques to the rigid body using the identified inertia and center of mass. (a) Estimating yaw torque using rotor acceleration and motor current; (b) Validation of identification flight parameters [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Rigid body parameters resulting from an identification flight, and the identified propulsion model

ℓ_2	$\boldsymbol{J}$	$\text{diag}\{2.58, 2.46, 4.32\} \cdot 10^{-2}$ kg m^2
	$\boldsymbol{r}_g$	$[4.28, -1.11, -11.1]^T \cdot 10^{-3}$ m
ℓ_1	$\boldsymbol{J}$	$\text{diag}\{2.54, 2.58, 5.46\} \cdot 10^{-2}$ kg m^2
	$\boldsymbol{r}_g$	$[4.01, -1.05, 8.64]^T \cdot 10^{-3}$ m

Note: Results obtained by batch least squares (l_2) and IRLS (l_1) do not differ significantly.

interpolator or by the path planner. The scalar thrust is given by $T = \|\boldsymbol{f}_i\|_2$ while the reference orientation $\boldsymbol{R}_d$ for the attitude controller is calculated from desired heading orientation $\boldsymbol{R}_{\psi_d}$ and computed thrust vector orientation $\boldsymbol{R}_{T_d}\boldsymbol{e}_3 = \frac{\boldsymbol{f}_i}{\|\boldsymbol{f}_i\|_2}$, such that $\boldsymbol{R}_d = \boldsymbol{R}_{\psi_d}\boldsymbol{R}_{T_d}$.

Based on the identified system parameters described above, the controller gains are derived from desired poles (in the negative complex half plane) of the closed-loop system. The chosen gains can be found in Table 3.

4.1.2 | External wrench estimation

To compensate for disturbances and estimate wind speed during flight, the hybrid wrench estimation approach by Tomić et al. (2017) is used. The estimator uses the control input, the dynamics model and the unbiased IMU only. The external wrench estimate $\hat{\boldsymbol{w}}_e = [\hat{\boldsymbol{f}}_e^T \ \hat{\boldsymbol{\tau}}_e^T]^T$ is obtained from

$$\hat{\boldsymbol{w}}_e = \begin{bmatrix} \int \boldsymbol{K}_I^f (m\boldsymbol{a} - T\boldsymbol{R}\boldsymbol{e}_3 - \hat{\boldsymbol{f}}_e)\mathrm{d}t \\ \boldsymbol{K}_I^\tau \left(\boldsymbol{J}\boldsymbol{\omega} - \int_0^t (\boldsymbol{\tau} + (\boldsymbol{J}\boldsymbol{\omega}) \times \boldsymbol{\omega} - \hat{\boldsymbol{\tau}}_e)\mathrm{d}t \right) \end{bmatrix}, \tag{13}$$

where $\boldsymbol{K}_I^f$ and $\boldsymbol{K}_I^\tau$ are filter gains, and $\boldsymbol{a} = \boldsymbol{R}^T(\ddot{\boldsymbol{p}} - g\boldsymbol{e}_3)$ is the acceleration measured by an accelerometer in the center of mass and expressed in the body frame. $\hat{\boldsymbol{f}}_e$ and $\hat{\tau}_e$ are the estimated external force and torque, also expressed in the body frame. Note that τ contains the term compensating the offset center of gravity. The estimation dynamics are shown to be $(s + K_I)\hat{\boldsymbol{w}}_e = K_I \boldsymbol{w}_e$. This estimator does not require translational velocity measurements. For the control input we assume hover conditions, and obtain the control thrust and torque using motor feedback. Therefore, the estimator will also capture modeling errors, aerodynamic drag, as well as the change of propeller thrust and torque with airspeed. For a complete and more detailed description of this framework the reader is referred to Tomić et al. (2018).

4.1.3 | Air density estimation

The thrust of a multirotor depends linearly on the air density ρ (see Equations (1) and (5)). A common assumption is that the air density is constant and known, and therefore, may be lumped in the rotor thrust and torque coefficients (Mahony et al., 2012). However, the air density changes depending on weather and altitude, that is, the atmospheric conditions pressure, temperature, and humidity. This can lead to a difference in thrust of more then 10%, which has to be compensated by the flight controller. Integrating a sensor measurement provides no guarantee for convergence of the thrust and includes a large uncertainty: Barometric sensor measurements are sensitive to wind and subject to large drifts. Temperature sensor measurements are affected by warm surrounding electrical components.

TABLE 3 Attitude and position controller gains

Attitude controller	$\boldsymbol{C}$	$\text{diag}\{245.0, 245.0, 180.0\}$
	$\boldsymbol{K}_\omega$	$\text{diag}\{17.0, 17.0, 12.0\}$
Position controller	$\boldsymbol{K}_p$	$\text{diag}\{2.0, 2.0, 4.0\}$
	$\boldsymbol{K}_d$	$\text{diag}\{1.6, 1.6, 4.0\}$

Hence, we have developed an air density estimator[5]. It does only rely on the state estimate (namely height and vertical velocity), which is available from the fusion (Section 4.2.2) of VO and IMU data, and on the desired trajectory. In addition to the air density, it also adapts to changing mass, for example, if a payload is collected or dropped.

We consider the air density ρ_m used for control allocation to be the real, a priori unknown, air density ρ subject to a multiplicative uncertainty $(1 + \varepsilon)$, such that $\rho_m = (1 + \varepsilon)\rho$. Inserting in the translational dynamics, Equation (2) yields also the expression $\hat{m} = (1 + \varepsilon)m$, which may be interpreted as an effective mass. The estimator for ε is basically an adaptation law defined as

$$\dot{\varepsilon} = -\gamma(\dot{\tilde{z}} + \lambda\tilde{z})(\ddot{z}_d - 2\lambda\dot{\tilde{z}} - \lambda^2\tilde{z} + g), \tag{14}$$

where $\gamma > 0$ is a design parameter, $\lambda > 0$ is the controller gain, and z, z_d with $\tilde{z} = z - z_d$ are the measured and the desired height, respectively. If the estimator is activated, (14) is integrated w.r.t. time and ε is used in the augmented position controller (see (12))

$$\boldsymbol{f}_i = (1 + \varepsilon)m(\ddot{\boldsymbol{p}}_d - \boldsymbol{K}_d(\dot{\boldsymbol{p}} - \dot{\boldsymbol{p}}_d) - \boldsymbol{K}_p(\boldsymbol{p} - \boldsymbol{p}_d) - g\boldsymbol{e}_3). \tag{15}$$

Finally, air density ρ or effective mass $\hat{m}$ are estimated via $\rho = \frac{1}{1+\varepsilon}\rho_m$ and $\hat{m} = (1 + \varepsilon)m$.

Figure 16 shows an experimental validation of the air density estimation. The measured atmospheric parameters of that day (2018/12/07) were 960 hPa and 20°C. It can be seen that the air density converges to the value 1.137 kg/m^3, which was calculated for comparison using the ideal gas law. A more accurate air density leads to a better estimate of applied thrust and torque, and therefore, also increases the quality of the external wrench estimation presented in Section 4.1.2.

[5]Patent submitted and pending.

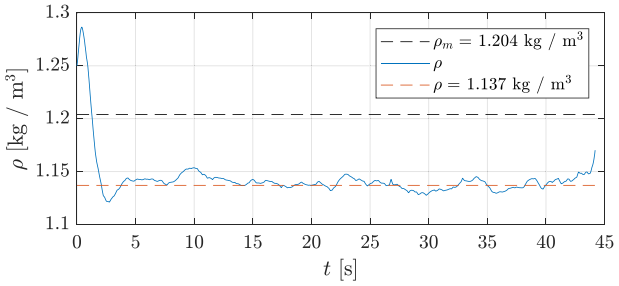


FIGURE 16 Estimated air density ρ (solid blue line). ρ_m is the model air density based on the sea level standard atmosphere (1,013 hPa, 20°C, black-dashed line) used in the control allocation. The sensor measurement for comparison is $\rho = 1.137$ kg/m^3 (red-dashed line) [Color figure can be viewed at wileyonlinelibrary.com]

4.2 | High level autonomy software

In this section, we describe the high level autonomy software which is running on-board the MAV. We describe the local state estimation of the vehicle, consisting of a multi-VO framework and a loosely coupled filter fusing the data of an IMU and the VO output. We then describe our framework for global 6D localization and dense 3D mapping that build on top of the local estimation. Finally, we present the motion planner of ARDEA which is using the processed point clouds for planning feasible and obstacle-free trajectories.

4.2.1 | Visual odometry

The VO is a crucial part for the state estimation and therefore, for the robustness of the system. Its task is to give an estimate of the camera motion based on the perceived images. Without pose estimates from the VO, the local navigation filter only integrates IMU measurements and therefore, would diverge within seconds, which makes the use of the MAV for autonomous tasks impossible. The presented VO estimates the relative transformation from one camera frame to another taken at different timestamps. The algorithm and setup is based on Hirschmüller, Innocent, and Garibaldi (2002) and Stelzer, Hirschmüller, and Görner (2012), where the reader is referred to for in-depth details. We assume that the scene is mainly static, which is a common assumption and also valid for most planetary exploration scenarios. Furthermore, we do not constrain the camera motions, it thus can be arbitrary. In contrast to other approaches that use motion priors or kinematic constraints derived for specific vehicles, we want our method to be as general as possible to be able to apply it on any robotic platform. We therefore, do not make any assumptions about the kinematics of the robot in the VO.

Figure 17 depicts the VO setup. After capturing and remapping the original camera images into virtual pinhole images (see Section 3.2) Adaptive and Generic Accelerated Segment Test (AGAST) features (Mair, Hager, Burschka, Suppa, & Hirzinger, 2010) are extracted in each of the left virtual pinhole images. With the dense depth map estimated via SGM (running on the FPGA), we can also retrieve depth information for the found features. As a result, we can track them in three dimensions for motion estimation. Features which do not have a valid depth value, for example due to occlusions, are discarded. In stereo vision setups, the uncertainty of their depth increases quadratically with the distance to the cameras. We therefore, ignore all features at depth values greater than 4 m for the motion estimation due to their potentially large errors.

Although three noncollinear 3D feature points are sufficient to calculate the translation and rotation of a camera's relative movement, it is advantageous to have more feature points to reduce the effect of noise, to thereby increase the estimation accuracy and to the improve rejection of outliers. After feature extraction, we search for correspondences between the current and previous image. Before a feature can be used for motion estimation, it has to pass two additional outlier rejection steps. Assuming the scene is static, one can expect that the relative distance $d_{\mathbf{sr}}$ of two points $\mathbf{s} \in \mathbb{R}^3$ and $\mathbf{r} \in \mathbb{R}^3$ does not change much from the previous frame $i-1$ to the current frame i. As a result the distance $d_{\mathbf{sr}}^{i-1}$ and $d_{\mathbf{sr}}^{i}$ in Equation (16) should equal up to small differences and measurement errors.

$$\begin{aligned} d_{\mathbf{sr}}^{i-1} &= \|\mathbf{s}^{i-1} - \mathbf{r}^{i-1}\|, \\ d_{\mathbf{sr}}^{i} &= \|\mathbf{s}^{i} - \mathbf{r}^{i}\|. \end{aligned} \tag{16}$$

The following outlier rejection step is based on an upper limit for the rotation angle of the camera. Finally, the remaining corresponding features $\mathbf{s}_k^i$ and $\mathbf{s}_k^{i-1}$, $k = 1, 2, \ldots, n$ can be used to estimate the camera motion. This is done by minimizing

$$E(\mathbf{R}, \mathbf{t}) = \sum_k \frac{1}{\sigma_k^2} \left(\mathbf{s}_k^{i-1} - \left(\mathbf{R}\, \mathbf{s}_k^i + \mathbf{t} \right) \right)^2. \tag{17}$$

Equation (17) uses a spherical error model, which is a rough approximation but has the benefit that the transformation can be calculated in closed form. After $\mathbf{R}$ and $\mathbf{t}$ are estimated, Chauvenet's criterion (Taylor, 1982) is applied to remove further likely false correspondences. Finally, the values for translation and rotation are optimized using an ellipsoid error model as described by Matthies and Shafer (Matthies et al., 2014). This is done by an initial guess derived from the previous spherical error model and with the reduced set of consistent correspondences.

Similar to Engel, Koltun, and Cremers (2018), Forster, Zhang, Gassner, Werlberger, and Scaramuzza (2017, Mur-Artal and Tardós (2017), and Wang, Schwörer, & Cremers (2017) we use keyframes for estimating not only the pose of the current camera frame relative to the previous frame, but also relative to a set of selected camera frames in the past. This increases the accuracy of the overall pose estimation. Every time an image is captured, its relative poses to all previous keyframes are calculated. The keyframe with the greatest residual error is replaced by the new image frame. In our current setup, we are using a fixed number of currently $n = 5$ keyframes. In terms of accuracy, computational time and memory consumption this value empirically lead to a very good performance in most of our use cases.

In our approach, a separate VO runs on each virtual pinhole stereo pair with a direct mapping between the virtual stereo camera and the

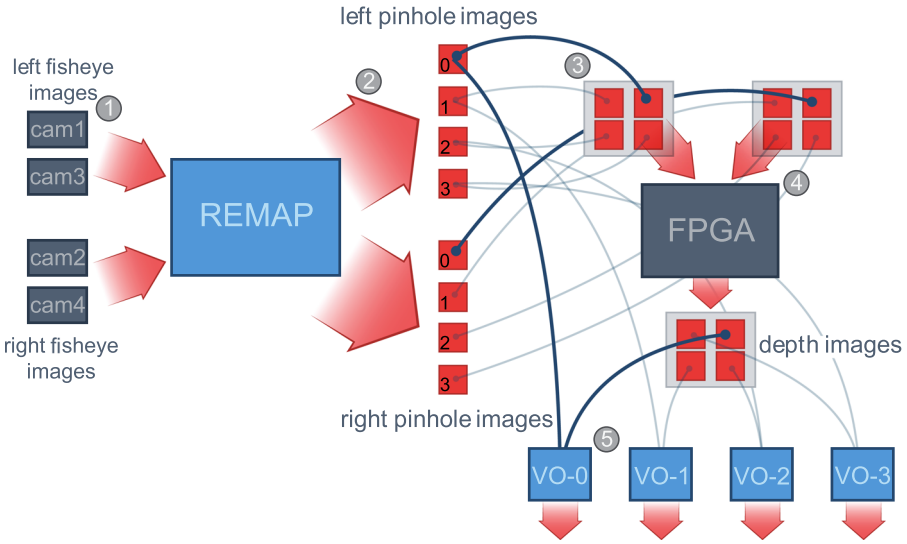


FIGURE 17 Basic camera and visual odometry setup. (1) Wide angle images are captured. (2) Those images are remapped into eight pinhole images. (3) All left and right images are grouped into one combined left and right image. (4) Left and right images are sent to FPGA for stereo processing, resulting in a depth map. (5) Each VO instance receives a pinhole image and the corresponding depth map [Color figure can be viewed at wileyonlinelibrary.com]

VO number as explained in Figure 4. Having independent VOs running on different areas of the field of perception allows to independently select different keyframes for each VO, as we show in Figure 18. Each point illustrates one keyframe and each arc indicates which reference frame the estimated, relative camera poses are expressed in. The resulting list of timestamps with their corresponding keyframes look different for each VO. For instance, while flying close to ground, features in the images of the downward-facing camera used by VO-0 are usually visible for much shorter intervals than features closer to the horizon, therefore VO-0 and VO-1 will select different keyframes. This results in additional robustness of the overall system. For more information, we kindly refer the reader to M. G. Müller et al. (2018).

4.2.2 | Local navigation filter

As outlined in Section 4.2.1, for each remapped virtual pinhole stereo camera an independent VO estimate is calculated. They are

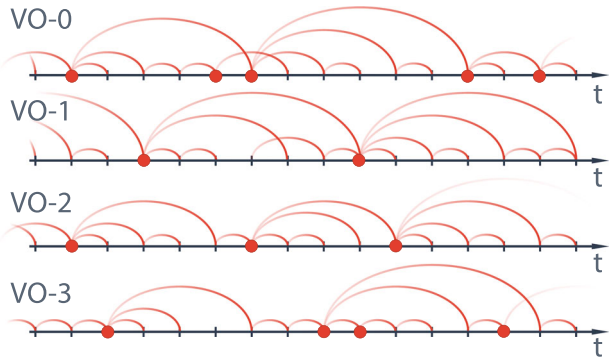


FIGURE 18 Keyframe handling of multiple VOs. Red dots illustrate keyframes, arcs indicate which reference frame the estimated, relative camera poses are expressed in. Samples without an arc indicate frames where pose estimation was not possible. Note that each VO selects different keyframes [Color figure can be viewed at wileyonlinelibrary.com]

combined with acceleration and angular rate readings from an IMU. Measurements from both sensors suffer from several sources of error. Among other things, noise and outliers of the VO measurements are caused by high motion dynamics or low texture in the images. The measurements of a microelectro mechanical system (MEMS) IMU mainly suffer from temperature change and mechanical stress of the sensor. This results in noise and time-varying biases superimposing measurements. Additionally, time delays due to transport and processing of the sensor data have to be taken into account. In Schmid, Ruess, Suppa, and Burschka (2012) and Schmid, Ruess, et al. (2014) an error state space Extended Kalman filter (EKF) was introduced to provide robust, nondelayed and accurate state estimates. An error state formulation has several advantages. Fast system dynamics are tracked by the Strapdown Algorithm (SDA) running at IMU sampling frequency of 500 Hz, whereas the Kalman filter steps are executed at a lower frequency (50 Hz), which is sufficient to track the slow error dynamics. A kinematics model is not needed, thus the algorithm can be easily used on different robotic systems. The direct $\mathbf{x}$ and indirect $\delta\mathbf{x}$ representation of the main state are defined by

$$\mathbf{x} = \begin{bmatrix} {}^{n}_{b}\mathbf{p}^T & {}^{n}_{b}\mathbf{v}^T & {}^{n}_{b}\mathbf{q}^T & {}^{b}\mathbf{b}_a^T & {}^{b}\mathbf{b}_\omega^T \end{bmatrix}^T,$$
$$\delta\mathbf{x} = \begin{bmatrix} {}^{n}_{b}\delta\mathbf{p}^T & {}^{n}_{b}\delta\mathbf{v}^T & {}^{n}_{b}\delta\boldsymbol{\phi}^T & {}^{b}\delta\mathbf{b}_a^T & {}^{b}\delta\mathbf{b}_\omega^T \end{bmatrix}^T, \tag{18}$$

where ${}^{n}_{b}\mathbf{p} \in \mathbb{R}^3$ is the position of the body frame (b-frame) relative to an earth-fixed, inertial frame (n-frame), ${}^{n}_{b}\mathbf{v} \in \mathbb{R}^3$ is the velocity, ${}^{n}_{b}\mathbf{q}$ the orientation represented as a unit quaternion, and ${}^{n}\mathbf{b}_a$ and ${}^{b}\mathbf{b}_\omega$ are the acceleration and angular rate biases of the IMU. For position, velocity, angular rate, and acceleration biases an additive error model is used, for example, $\mathbf{p} = \hat{\mathbf{p}} + \delta\mathbf{p}$. For the orientation a multiplicative error model, ${}^{n}_{b}\mathbf{q} = {}^{n}_{b}\hat{\mathbf{q}} \otimes {}^{n}_{b}\delta\mathbf{q}$ is used, where the error rotation vector ${}^{n}_{b}\delta\boldsymbol{\phi} \in \mathbb{R}^3$ is represented by the error quaternion ${}^{n}_{b}\delta\mathbf{q}$.

The transport and processing of images to calculate the VO estimates introduces time delays which would negatively influence the accuracy and robustness of the state estimation and hence the

stability of the system. Therefore, they have to be compensated. Hardware triggers are used to define the exact timestamp, when an image was exposed. Additionally, each time an image is triggered, the current state $\bar{\mathbf{x}}_k$ is extended by a substate $\mathbf{x}_{\text{aug}}$

$$\bar{\mathbf{x}}_k = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{\text{aug}} \end{bmatrix}. \tag{19}$$

The substate $\mathbf{x}_{\text{aug}}$ depends on the measurement equation of the sensor used. In the case of VO readings, the measurement equation is in the form of

$$\mathbf{h}_{k_1,k_2} = \mathbf{h}\left({}^{n}_{b}\mathbf{p}_{k_1}, {}^{n}_{b}\mathbf{q}_{k_1}, {}^{n}_{b}\mathbf{p}_{k_2}, {}^{n}_{b}\mathbf{q}_{k_2}\right). \tag{20}$$

and $\mathbf{x}_{k,\text{aug}} = \begin{bmatrix} {}^{n}_{b}\mathbf{p}_k \\ {}^{n}_{b}\mathbf{q}_k \end{bmatrix}$. The subscripts k_1 and k_2 refer to the start time t_1 and end time t_2 of the VO measurement, that is t_1 is the timestamp of the keyframe, relative to which the motion at time t_2 has been estimated. Since each VO runs independently, measurements from different VOs with identical end times t_2 can have different start times t_1. Each time a hardware trigger arrives, the main state and the covariance matrix are augmented. The main state is augmented by the current pose and the covariance matrix by the submatrix representing the uncertainty of the current pose. In addition, the equation for system propagation has to be adapted. This provides the filter with all information it needs to process the measured value when it arrives and correct the current state, including the augmentations. Figure 19 shows an overview of the filter design and the connection to the VOs and IMU.

The VO is the only sensor complementing the readings from the IMU. Therefore, outliers in the VO will directly have a negative impact on state estimation if not detected correctly. Some precautions to detect outliers are already built into the VO itself. But not all cases can be directly caught in the VO. Additional knowledge about the motion, gained by IMU readings, can be exploited to further reduce the likelihood to incorporate undetected outliers from the VO in the filter. Based on the actual measurement from the VO, the predicted measurement and their corresponding covariances, a distance measure, the Mahalanobis distance (Wu, Chen, Yang, & Chen, 2016), is calculated. If its value is above a suitable threshold, the measurement is considered to be an outlier and rejected. The value of the threshold depends on the number of rows in the measurement equation and the significance level α and can be selected from a precalculated chi-square table. For 6 DOF and a significance level of $\alpha = .99$, the threshold is 16.8. In this way, the influence of outliers in the VO is minimized.

4.2.3 | Global 6D localization and 3D mapping

For global localization and mapping, we employ our 6D SLAM framework introduced in Schuster, Brand, Hirschmüller, Suppa, and Beetz (2015, 2018). It enables efficient online and on-board single- and multirobot global localization and mapping by building upon the estimates from the local navigation filter and complementing them with additional intra- and inter-robot loop closure constraints. Combing local and global estimation methods allows us to get the best of both worlds: Fast local state estimates from the navigation filter that are required for stabilization and control of our highly dynamic robot as well as online global estimates required for consistent mapping, path planning, exploration as well as multirobot coordination.

To create dense 3D maps we employ a submapping technique. It allows us to efficiently handle the high-bandwidth depth data generated via the FPGA-based stereo matching on the images from ARDEA's wide-angle camera system presented in Section 3.2. As a first step, we aggregate the dense stereo data along the trajectory estimated by the local navigation filter (Section 4.2.2). As its estimates are locally stable but globally subject to drift, we partition the aggregated data into partial maps of limited size and uncertainty, so-called submaps. A submap, anchored by the gravity-aligned pose of its origin, contains two different, application-dependent representations for its 3D data that we visualized in Figure 20: Colored point clouds at a resolution of 5 cm and probabilistic voxel space at a resolution of 10 cm. Point clouds are fast to aggregate and constitute a suitable 3D model for visualization of the environment and, in the future, can serve as input for semantic segmentations and geometry-based map matching methods (Brand, Schuster, Hirschmüller, & Suppa, 2015). We employ the freely available *OctoMap* library for a memory-efficient representation of a 3D voxel space (Hornung et al., 2013). The probabilistic aggregation of data from multiple measurements is computationally more expensive, hence the lower resolution compared to the pointcloud representation, however, it allows to deal with sensor noise and changing parts in the environment. Furthermore, this representation explicitly distinguishes

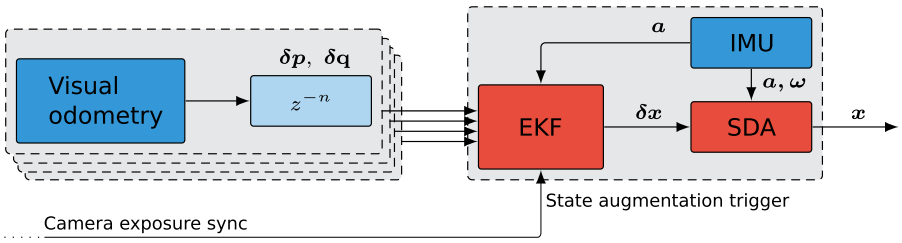


FIGURE 19 Local navigation filter design. The direct system state $\mathbf{x}$ is calculated at a high rate by the SDA using acceleration and gyroscope measurements ($\mathbf{a}$, ω) coming from the IMU. Relative and time delayed pose measurements ($\delta\mathbf{p}$, $\delta\mathbf{q}$) are used in the EKF to calculate the state errors $\delta\mathbf{x}$ at a lower rate and are immediately used for state correction [Color figure can be viewed at wileyonlinelibrary.com]

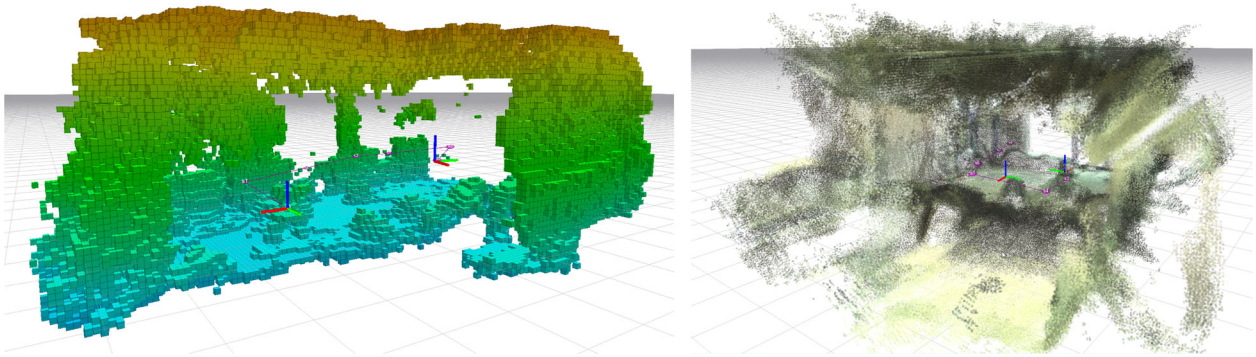


FIGURE 20 Three-dimensional map representations: Probabilistic voxel grid (height colored, left) and pointcloud (uncalibrated camera colors, right) maps of our laboratory created online and on-board ARDEA during a demonstration of a waypoint flight [Color figure can be viewed at wileyonlinelibrary.com]

between *unknown*, *occupied*, and *free* space, information that is crucial for obstacle avoidance, path, and exploration planning algorithms. In the experiments presented in Section 6.1, new submaps were triggered whenever within a submap, the standard deviation of the robot's position as estimated by the local navigation filter exceeded 0.1 m or the accumulated traveled distance was above 2.0 m. These thresholds limit the errors within the individual submaps caused by filter drift and restrict their size to limit their memory and processing time requirements in postprocessing steps and multirobot data exchange. Whenever a new submap is triggered, we switch the frame of reference of the local navigation filter (Section 4.2.2), which is implemented as a *local reference filter* (Schmid, Ruess, et al., 2014), into the gravity-aligned submap origin. This frame switching allows to maintain long-term consistency and numerical stability within the filter as well as a more accurate integration of the filter's estimates into the overlying SLAM graph (Schmid, Ruess, et al., 2014; Schuster et al., 2015).

For online global pose and map estimation, we optimize a pose graph: Submap origins, robot, and landmarks poses are modeled as nodes in an undirected graph that is constructed in an incremental fashion at runtime. These nodes are connected via estimate and measurement constraints, represented as edges that are weighted according to their respective Gaussian uncertainties. The sparse structure of the graph reflects the (in)dependencies of the underlying optimization problem. We compute a maximum a-posteriori solution on loop closures via the *iSAM2* algorithm (Kaess et al., 2012) for iterative least-squares error minimization that is implemented in the freely available open-source *GTSAM 3.2.1* library (Dellaert, 2015).

Whenever new submaps are created, we add their origins as nodes to the graph and connect them via the filter estimates for its respective switch of reference frame. In Figure 21 we sketched a SLAM graph for a multirobot scenario. Loop closure constraints result from the observation of static landmarks or the marker-based detection of other robots. They connect robot poses estimated by the local reference filter w.r.t. their respective submaps. To keep the graph small, robot poses are only added where needed to connect other measurements. Once inter-robot measurements are available, it is straightforward to include the nodes and edges from other robots to create a joint graph for multirobot estimation. In Section 6.1 we present a demonstration of our multirobot mapping system in a heterogeneous team consisting of our aerial robot ARDEA and the planetary exploration rover LRU (Schuster et al., 2017). Combining local navigation filters, one per robot, with pose graph optimization leads to a small and sparse graph, allowing fast incremental online optimization steps. The SLAM graph thereby is independent from high-frequency measurements and filter-internal states. This is particularly important for systems like ARDEA with its four key frame-based visual odometries (Section 4.2.1). As all of their estimates are fused in the local navigation filter, the SLAM graph does neither increase in size nor complexity by adding further high-frequency measurements or estimates like, in this case, additional visual odometries. Furthermore, in multirobot systems, high-frequency measurements and high-bandwidth depth data are processed locally on each robot in a distributed fashion and then only transferred and combined in their aggregated and compacted forms.

We periodically compose global dense 3D maps from the submaps, as shown exemplarily in Figure 20 for single and in Section 6.1 for multirobot experiments. This is done by arranging them according to the latest graph SLAM estimates for their origins and merging their 3D representations.

4.2.4 | Motion planning

This software component tackles the problem of global motion planning that is, generating feasible point-to-point motions within a global map and sending them to the controller as reference trajectories. Our solution is comprised of a custom, standalone motion planning library, and middleware interface which provides the communication between both the real-time and ROS middlewares and the library API. This point-to-point formulation is convenient because it is modular, and therefore, the scope is easily broadened or narrowed.

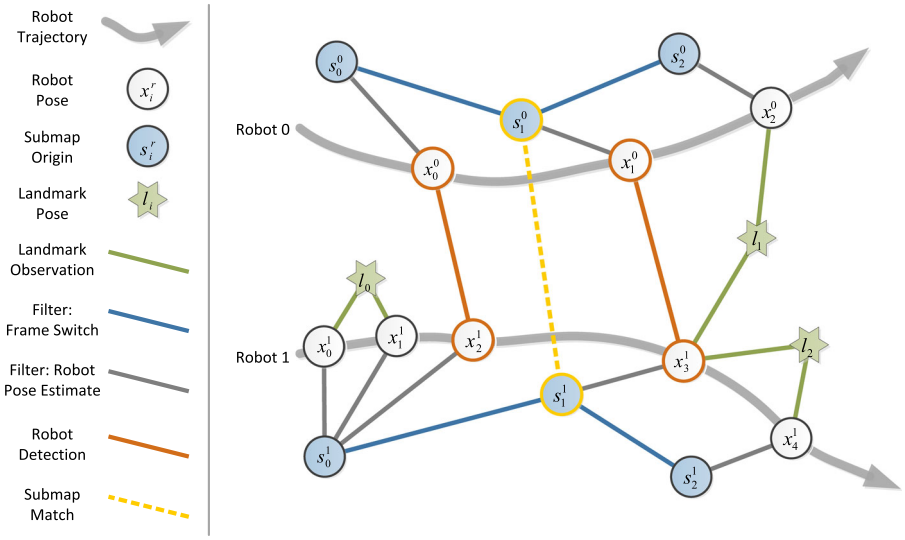


FIGURE 21 Sketch of multirobot simultaneous localization and mapping graph integrating the estimates of local reference filters as well as loop closure constraints in form of landmark observations, robot detections, and submap matches [Color figure can be viewed at wileyonlinelibrary.com]

Problem formulation

We formulate trajectory generation as a nonlinear program to minimize a cost Γ which is a function of some free parameters $\mathbf{p}$ describing the path of the system $\mathbf{T}(\mathbf{p}, \mathbf{t})$ and its time derivatives (a trajectory). This is an inverse dynamics-based approach where the states along a trajectory are independently parameterized and the actuations to achieve them arise from the system equations of motion shown in Equation (2), (3), and (4). We denote ARDEA's pose as $\mathbf{r}(\mathbf{p}, t) \in \mathbb{R}^4$ for the four flat outputs (x, y, z, ψ) and the state as $\mathbf{y}(\mathbf{p}, t) \in \mathbb{R}^{j\times 4}$, comprising a pose and its time derivatives,

$$\begin{aligned}\mathbf{y}(\mathbf{p}, t) &= \begin{bmatrix} \frac{d^0}{dt^0}\mathbf{r}^T(\mathbf{p}, t) \\ \vdots \\ \frac{d^j}{dt^j}\mathbf{r}^T(\mathbf{p}, t) \end{bmatrix} \\ &= \begin{matrix} \frac{d^0}{dt^0} \\ \vdots \\ \frac{d^j}{dt^j} \end{matrix} \begin{bmatrix} r_x(\mathbf{p}_x, t) & r_y(\mathbf{p}_y, t) & r_z(\mathbf{p}_z, t) & r_\psi(\mathbf{p}_\psi, t) \\ & \vdots & & \\ r_x(\mathbf{p}_x, t) & r_y(\mathbf{p}_y, t) & r_z(\mathbf{p}_z, t) & r_\psi(\mathbf{p}_\psi, t) \end{bmatrix}. \end{aligned} \tag{21}$$

The rotational part is expressed as an angle in radians and rotations in SO(3) as unit quaternions $\mathbf{q}$ using the axis-angle conversion for numerical robustness. For brevity it is assumed in the rest of this section that $\mathbf{y}(\mathbf{p}, t)$, $\mathbf{r}(\mathbf{p}, t)$, and so forth, are functions of the parameters $\mathbf{p}$ and time t.

The optimization problem is formulated as,

$$\begin{aligned} \min_{\mathbf{p}} \quad & \Gamma(\mathbf{p}) = \int_{t_0}^{t_{n_{\text{via}}}} f(\mathbf{y})\, dt \\ \text{subject to:} \quad & \mathbf{c}_{\text{ineq}}(\mathbf{y}) = \begin{bmatrix} \mathbf{c}_{\text{vel}}(\dot{\mathbf{r}}) \\ \mathbf{c}_{\text{act}}(\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}) \\ \mathbf{c}_{\text{col}}(\mathbf{r}) \end{bmatrix} \leq 0 \\ & \mathbf{y}(t_0) = \mathbf{y}_{t_0}, \quad \mathbf{y}(t_{n_{\text{via}}}) = \mathbf{y}_{t_{n_{\text{via}}}} \end{aligned} \tag{22}$$

The objective function $\Gamma(\mathbf{p})$ and the inequality constraints encode the equations of motion which are inherently coupled. Three vector constraints make up $\mathbf{c}_{\text{ineq}}$: upper and lower bounds on the robot velocity $\mathbf{c}_{\text{vel}}$ arising from the limits of the VO described in Section 4.2.1 as well as bounds on the motor speeds $\mathbf{c}_{\text{act}}$, and collision constraints $\mathbf{c}_{\text{col}}$. Because angular velocities are known to cause strong motion blur, the upper bounds must be chosen according to worst case light conditions, that is with the slowest allowed camera shutter speed. The initial and final states are not a part of the optimization problem but are fixed in Equation (26). Equation (22) is implemented such that the trajectory is discretized with a fixed number via points n_{via}, where $t \in [t_0 \;\; t_{n_{\text{via}}}]$ and must be feasible according to the nonlinear inequality constraints $\mathbf{c}_{\text{ineq}}$ at each discrete via point. We solve Equation (22) using the sequentially least-squares quadratic programming (SLSQP; Kraft, 1988) implementation from the open source NLopt library (Johnson, 2008). The algorithm is gradient-based and the numerical gradients for Γ and $\mathbf{c}_{\text{ineq}}$ are solved by perturbing the parameters $\mathbf{p}_i + \eta$ and computing the forward finite differences.

Cost metric

For the optimization objective Γ we make use of a weighted, squared sum of the linear and angular accelerations along the discrete trajectory,

$$\Gamma_{\text{acc}}(\mathbf{p}) = \mathbf{w}^T \cdot \sum_{t=t_0}^{t_{n_{\text{via}}}} \begin{bmatrix} \ddot{\mathbf{r}}^T\ddot{\mathbf{r}} \\ \dot{\boldsymbol{\omega}}^T\dot{\boldsymbol{\omega}} \end{bmatrix}. \tag{23}$$

Minimization of the rigid body accelerations is a practical choice of cost metric for our proposed exploration scenarios. The justification is threefold: (a) computing the accelerations is computationally expedient, (b) the resulting motions are ideal for collecting sensor data, (c) acceleration is typically quadratic in the optimization parameters, thus descent to a local minimum is fast. Many state-of-the-art motion planning and control methods for MAVs make use of more complex metrics. Time minimization for aggressive or agile flight (Liu, Mohta, Atanasov, & Kumar, 2018; Richter et al., 2016) has been an area of active research since the advent of MAVs. Our proposed objectives

for planetary exploration would require maximal robustness and image acquisition quality, therefore, precluding the need for aggressive flight. Sensor-based quantities such as cinematic effect (Nägeli et al., 2017) or mapping of a point of interest (Yoder & Scherer, 2016) are also of increasing interest. While such planning metrics could be useful for optimal data gathering, they are out of the scope this study.

Inverse dynamics procedure

The actuation constraints $\mathbf{c}_{\text{act}}$ are realized as upper and lower bounds on the required motor speed to achieve the flat outputs as opposed to simple limits on the system acceleration. While this approach is more computationally intensive, it ensures that the planned trajectories are feasible w.r.t. the system dynamics as described in Section 4.1.1 and Equation (1). To obtain the motor speeds given the state as flat outputs, $\mathbf{y}$, the following procedure is used:

$$\begin{aligned} \mathbf{f}_I &= m\ddot{\mathbf{r}} - m\mathbf{f}_{\text{g}}, \\ \mathbf{q}_{BI} &= \mathbf{q}(\hat{\mathbf{z}}_B, \hat{\mathbf{f}}_I)\mathbf{q}(r_\psi), \end{aligned} \tag{24}$$

where $\mathbf{f}_I$ is the external force vector in the inertial frame, $\mathbf{q}_{BI}$ is the unit quaternion rotated from an inertial frame to ARDEA's body frame, $\mathbf{q}(\hat{\mathbf{z}}_B, \hat{\mathbf{f}})$ is the rotation from the body z-axis to the force vector, and $\mathbf{q}(r_\psi)$ is the yaw (heading) represented as a unit quaternion. $\mathbf{q}_{BI}$ is differentiated twice using 3-point finite differences and the resultant quaternion rates are converted to angular velocities ω and accelerations $\dot{\omega}$. Then the external torque τ is

$$\tau = \mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega + m\left(\mathbf{r}_{\text{CoG}} \times \mathbf{q}_{BI}^{-1}\mathbf{f}_g\right). \tag{25}$$

Given the external torque and force, the motor speeds can be calculated by solving the linear inverse problem as described in the control allocation paragraph in Section 4.1.1.

Parameterization

The trajectories are parameterized with clamped, uniform Basic Splines (De Boor, 2001) of order $k = 6$. The curves $\mathbf{S}(\mathbf{p}, \tau)$ are a function of vertices $\mathbf{p}$ and an independent knot parameter τ, as well as the basis functions $\mathbf{N}(\bar{\tau}, \tau)$ and a knot vector $\bar{\tau}$. We equate the independent parameter τ with time t and represent each independent flat output as a single DOF curve, such that a trajectory is composed of four splines, defined by a matrix of vertices. The boundary conditions, which are the input to the point-to-point problem, are handled such that each curve

$$\begin{aligned} &\frac{d^r}{d\tau^r}\mathbf{S}(\mathbf{p}, t) = \frac{d^r}{d\tau^r}\mathbf{N}(\bar{\tau}, t)\begin{bmatrix}\mathbf{p}_0 \\ \mathbf{p} \\ \mathbf{p}_f\end{bmatrix}, \\ &\frac{d^r}{d\tau^r}\mathbf{S}(\mathbf{p}, t) \in \mathbb{R}^{n_{\text{via}}}, \quad \frac{d^r}{d\tau^r}\mathbf{N}(\bar{\tau}, t) \in \mathbb{R}^{n_{\text{via}} \times n_{\text{vts}}}, \\ &\begin{bmatrix}\mathbf{p}_0^T & \mathbf{p}^T & \mathbf{p}_f^T\end{bmatrix}^T \in \mathbb{R}^{n_{\text{vts}}} \end{aligned} \tag{26}$$

is solved for some vertices $\mathbf{p}_0$ and $\mathbf{p}_f$ given $\frac{d^r}{d\tau^r}\mathbf{S}$ at t_0 and t_f for the desired number of time derivatives $r = 0, 1, ..., n_{\text{fbcs}} - 1$. Position, velocity, and acceleration are given by the desired waypoints, additionally jerk and snap are set equal to zero[6]. This ensures C^2 continuity in acceleration at the endpoints. The free vertices $\mathbf{p}$ are the optimization parameters.

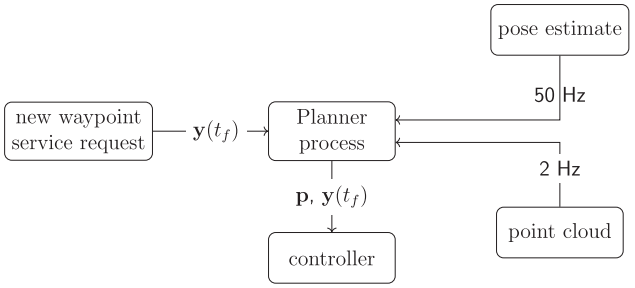


FIGURE 22 Schematic of the planning software integration. The planner process runs continuously in an idle state, generating only when a new waypoint is received via acyclic trigger

Collision constraints

The trajectories are constrained by the system dynamics and camera properties, as well as that they should be collision free. To ensure that a path traverses only free space, the planning software receives as input a global probabilistic occupancy grid represented as an Octomap (Hornung et al., 2013). Given a trajectory $\mathbf{T}$, at each discrete via point t the Octomap is queried for the occupancy probability at $\mathbf{r}(t)$.

Random search

While the gradient-based, nonlinear optimization is a powerful tool for finding smooth, feasible, and locally optimal solutions given the constraints and cost metric, it is highly dependent on an initial guess and susceptible to existing local minima. Similar to the combinatorial method described in Stoneman and Lampariello (2016) we run an initial, RRT-like coarse search to seed the optimization with a feasible initial guess. The coarse search is a modified RRT algorithm which samples position and velocity of the flat outputs from a uniform distribution. The edges are composed using the point-to-point spline method as described above, where the accelerations of each DOF are minimized using a bounded, Eigen-based quadratic program (Guennebaud, 2017; Guennebaud & Jacob, 2010).

Trajectories and software integration

A schematic of the planner software inputs and outputs is shown in Figure 22. The motion planner runs continuously and actions can be triggered via acyclic service requests. A planning action receives as input a desired waypoint state(s) and generates trajectories from the robot's estimated current state to the waypoints(s). When a feasible solution is found the resulting spline vertices and motion duration are sent to the controller which then queries the reference each tick.

[6]Jerk and snap are the third and fourth time derivatives of position, respectively.

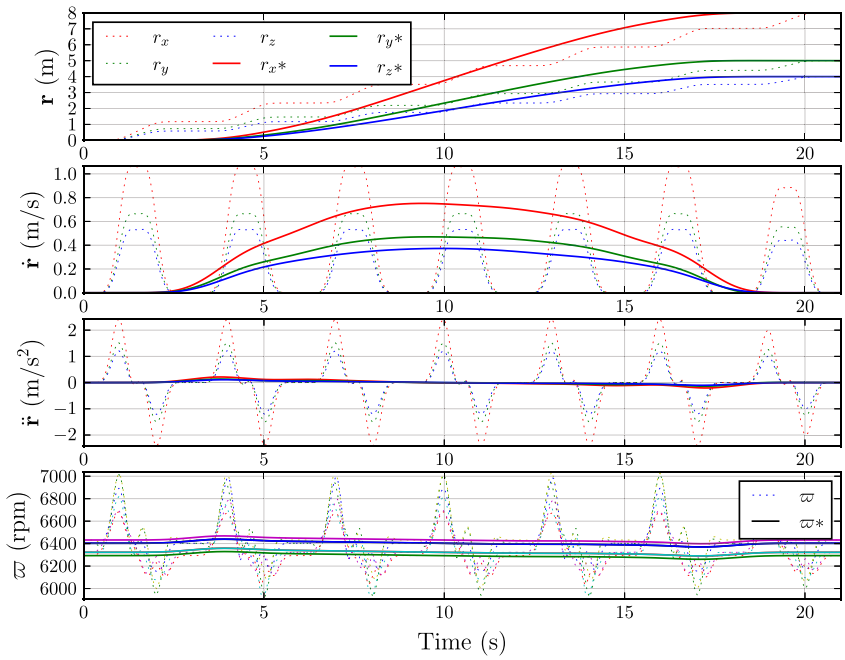


FIGURE 23 A typical reference trajectory. The dotted lines show the coarse solution resulting from the RRT-like random search, and the solid lines show the result after this initial guess has been *smoothed* by the optimization [Color figure can be viewed at wileyonlinelibrary.com]

An example trajectory from one stationary waypoint to another is shown in Figure 23. The result of the coarse search which is locally optimal and satisfies the motion constraints at each edge is provided to the optimization algorithm as an initial guess. These typically jerky, piece-wise trajectories are then smoothed according to the cost metric.

5 | SKILLS

Skills are a well-known concept in robotics. Many different definitions of the term "skill" exist in the literature. Some focus on the intellectual capability of problem solving (Sussman, 1973), some on the physical abilities of the motor system (Peters, Kober, Mülling, Nguyen-Tuong, & Kroemer, 2012), and some on all functionalities that may be implemented in a state machine (Steinmetz & Weitschat, 2016).

We define skills as modular sets of perceptual, computational, and dynamical capabilities that the MAV possesses. Skills can be computational operations or actions that the vehicle executes. They may have a number of static parameters or dynamic variables from input data (e.g., a measurement or state estimate). The outcomes of the skills can be the result of a computation, an action, or both. We follow the definition from (Ogasawara, Kitagaki, Suehiro, Hasegawa, & Takase, 1993) where skills are defined as primitives which execute a combination of functions. We adopt this scheme by allowing a skill to have only numeric outputs instead of actions.

The introduction of skills enables an easy access to basic and more complex capabilities of the MAV. The set of available skills allows an operator to define how a task should be solved in a structured way. For clarity, we divide a mission into tasks, which may use a defined set of skills, see Figure 24. This increases modularity, allows reactive changes of the task sequence,and easy definition of new missions, which can inherit functionalities (in the form of tasks and skills) from existing missions. Another advantage of the skill concept is that its level of abstraction can be represented well by a state-of-the-art state machine, such as RAFCON (Brunner, Steinmetz, Belder, & Dömel, 2015).

In the following, we describe the most important skills of ARDEA. We first treat basic skills inherent to any modern MAV. Then, we introduce more advanced skills, which lead to a higher level of autonomy, required for future a planetary mission.

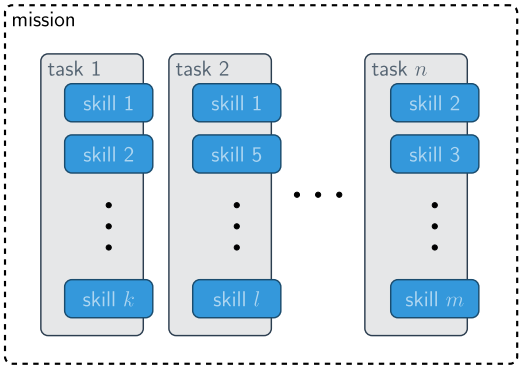
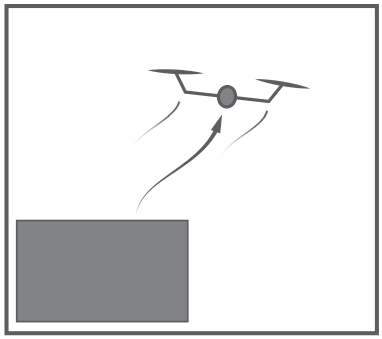


FIGURE 24 General concept of breaking down a mission into tasks and skills. A mission is defined by the fulfilling of a number of tasks, whereas tasks are defined by a set of skills [Color figure can be viewed at wileyonlinelibrary.com]

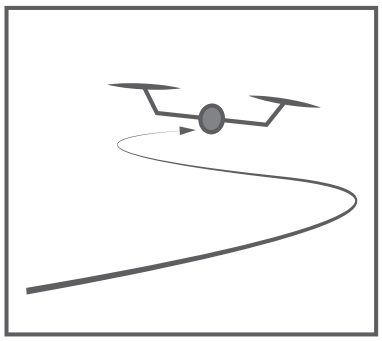
5.1 | Takeoff skill



Parameter	Value
Start thrust	28 N
Thrust slope	14 N s^{-1}
Takeoff waypoint	[0., 0., −0.5] m
Takeoff duration	1.5 s

Autonomous takeoff is required for every mission of a flying robot and consists of two phases. During the first phase, the thrust of ARDEA is increased until a desired takeoff thrust is reached. Then takeoff to a predefined waypoint is performed. The skill is parameterized by the initial thrust, the slope of a linear thrust ramp, a waypoint, and the duration. In general, fast takeoffs are preferred to reduce the influence of the ground effect, especially if obstacles are close by. Our usual starting point in a heterogeneous robot team is a platform on the back of LRU (Schuster et al., 2017). The scientific sensor unit of the rover, which also includes several cameras, is close to the starting point of ARDEA. Therefore, the takeoff waypoint is chosen such that collisions with the camera system of the LRU are avoided. In contrast, when using a visual sensor for state estimation, abrupt and fast movements may lead to lost features or motion blur. Hence, the chosen parameters are a trade-off and they were derived empirically for a smooth takeoff.

5.2 | Fly-to-waypoint skill



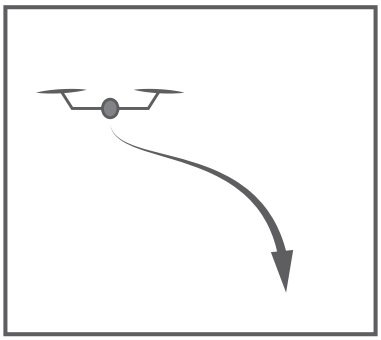
Parameter	Value
Max. linear velocity	0.5 m s^{-1}
Max. yaw velocity	0.6 rad s^{-1}
Vicinity value	0.3 m

The fly-to-waypoint skill enables ARDEA to track trajectories to waypoints defined on-the-fly. The skill is implemented as a module in the custom controller described in Section 4.1.1. For rest-to-rest maneuvers the skill uses fifth-order polynomials (M. W. Müller et al., 2015) to interpolate between an incoming waypoint and ARDEA's on-board state estimation. For maneuvers which require collision avoidance or guarantees of feasibility, it can receive and evaluate spline vertices from the motion planner (Section 4.2.4). The fly-to-waypoint skill can be triggered directly by a state machine with a waypoint or by the motion planner, determining whether the interpolator or spline evaluator is used. Both the polynomial interpolator and the spline evaluator are configured for the flat output space, so that waypoints are four DOF poses and the spline vertices are matrices in $\mathbb{R}^{n_{\text{vts}}\times 4}$.

The polynomial interpolator constrains the velocity along the trajectory by setting the maneuver duration given the maximum distance of the four independent DOFs. Given a desired goal waypoint $\mathbf{r}_g$, and a maximum velocity $\dot{\mathbf{r}}_{\max}$, the duration T of the trajectory can be computed as in M. W. Müller et al. (2015), $T = \frac{15}{8}\frac{\max(\mathbf{r}_g - \mathbf{r}_0)}{\dot{\mathbf{r}}_{\max}}$, where $\mathbf{r}_0$ is the current position. The parameters of the interpolation are thus the maximum allowed linear and yaw velocities or a fixed duration T. The parameters were derived empirically for flights with a moderate velocity to avoid motion blur and increase state estimation accuracy.

After a polynomial has been interpolated, or upon receiving spline coefficients from the motion planner, the resultant trajectory is evaluated at each control tick and the reference is tracked by the cascaded position and attitude controller. A waypoint is considered reached when ARDEA is within a defined vicinity around the waypoint.

5.3 | Landing skill

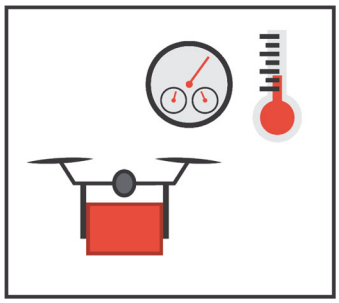


Parameter	Value
Max. linear velocity	0.5 m s^{-1}
Max. yaw velocity	0.6 rad s^{-1}

The landing skill consists of a waypoint flight and the shutdown of the engines as soon as the MAV has landed. Depending on the structure of the terrain at the landing spot, the cameras of the MAV might be too close to the ground to obtain any depth information after landing. As a result, new pose estimates cannot be

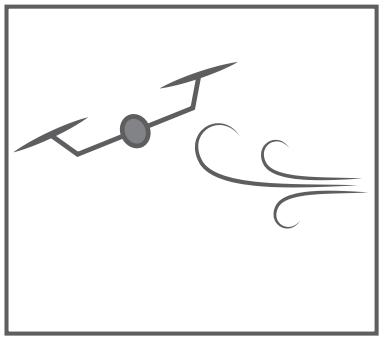
calculated by the VO. The poses from the VO contain the only information that is used to compensate the drift caused by the integration of the IMU data in the state estimation module. Therefore, it is important to notify the state estimation module after a successful landing. Skipping this step, could result in a nonpredicable behavior of the MAV, like takeoff into a random direction. In addition, noisy measurements might be accumulated in the map due to a drifting pose estimate. Similar to the fly-to-waypoint skill, the landing skill takes the desired landing waypoint and velocity as input parameters.

5.4 | Air density and payload estimation skill



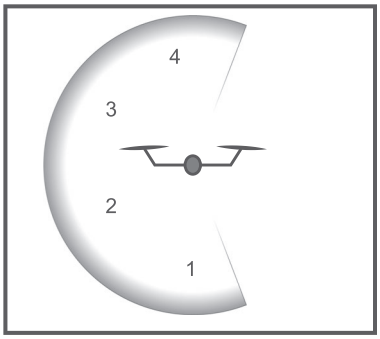
An estimate of the air density is important especially for planetary missions. It can not only be used for increased flight performance, but also for meteorological and scientific measurements. The air density estimation skill implements the method presented in detail in Section 4.1.3. It can be activated via a service call—during takeoff to estimate the air density, or if a payload is collected or dropped—and deactivated afterward to avoid adaptation to external disturbances.

5.5 | Wind estimation



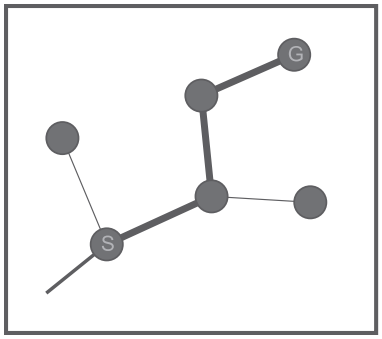
As mentioned in Section 4.1.2, external disturbances such as wind can be modeled in the control software and estimated. This can help to improve stable flight in missions where strong disturbances are expected or enable MAVs to act as a flying wind sensor to characterize complex airflow scenarios, for example, next to wind parks. Sensors which measure wind velocities, such as pitot tubes or anemometers have the drawback that the complex airflow around the MAV caused by the propulsion system directly influences the sensor readings and thus can render them useless (Tomić et al., 2016). Isolating the wind sensor from the propulsion airflow makes the MAV design and sensor placement a challenging task. We estimate this quantity by using the aerodynamics properties of the propellers and the motor current measured by the ESCs. This not only simplifies the system design by removing the need for an additional sensor, but also mitigates this issue.

5.6 | Modular VO setup skill



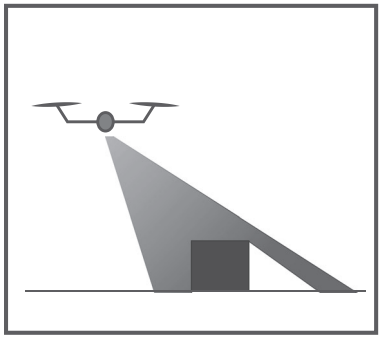
Each VO can be activated and deactivated on-the-fly and independently of the state estimation module. As such it only takes a VO ID and the respective on/off status flag as parameters. If this skill stops a VO, the local navigation filter does not receive any information of this particular VO instance anymore. Being able to switch on and off sensors in the state estimation is one advantage of a loosely coupled filter approach. In case the filter receives no VO measurements at all, it will propagate its current state using the IMU data. In general, this should be avoided, at least for longer periods of time, since the filter will diverge. However, being able to switch off VO(s) can be of great benefit in specific mission environments. Dynamic scenes, for example, can lead to wrong VO measurements and, if image motion is caused by large moving objects, even to erroneously small covariance estimates, which may not be rejected by the filter. For known moving objects within the FOV of one VO, it is possible to disable the respective VO temporarily. In addition, a single VO or several VOs can be deactivated to reduce the computational burden and energy consumption of the overall system. In particular, this is reasonable for flat outdoor scenes where not all VOs are necessary.

5.7 | Waypoint planning skill



The waypoint planner maintains a graph of waypoints in ARDEA's global map frame and can traverse its graph to find optimal sequences from one waypoint to another. The waypoint planning skill provides the input to either the trajectory planner in Section 4.2.4 or the polynomial interpolator, Section 5.2, as depicted in Figure 9. It can perform three different actions when triggered, construct a graph from a list or from waypoints defined dynamically for example, from an exploration task, plan a sequence through the graph from a start to a goal waypoint, or send the next waypoint in a planned sequence. The graph allows each node to have multiple child waypoints, but just one parent waypoint. Optimal sequences of waypoints are calculated using the well-known Dijkstra algorithm (Dijkstra, 1959).

5.8 | Depth estimation skill



The depth estimation skill estimates the average distance from the MAV to a particular region of interest. To obtain depth information, the skill uses the stereo vision setup. The region of interest is given by a polygon and can be passed as input, in form of a list, to the skill. The polygon is defined as a set of 3D points with respect to an arbitrary frame and projected into the image plane of the respective camera. To project the region of interest, the skill also takes as input the current position of the MAV and the polygon reference frame. With that information it calculates the relative transformation from polygon points to camera. The skill measures the depth values in the region of interest, averages them and provides the result as output. Additionally, the depth estimation skill also calculates the average depth value, if a planar surface would be present. The option is just available, if the defined polygon spans a flat surface. This can be used to estimate if an obstacle is present in the region of interest, if dealing with a more structured environment.

It can also be useful for estimating the current height of the vehicle. To do this, the reference frame of the polygon should be the IMU frame, to move with the vehicle. The polygon can then be defined as an area below the MAV. Since the polygon is defined in 3D space, the height of the vehicle should be roughly known beforehand. This can be done with height estimations, which where done before or by averaging all depth values in the depth image of the down looking camera to get a first estimate. The measured depth d_{pc} is based on a pinhole camera model. Therefore, the measured values are the distance projected on the principle axis of the camera. To measure the distance between camera and 3D point, the angle ϕ between principle axis and ray to 3D point has to be taken into account. The angle is calculated by

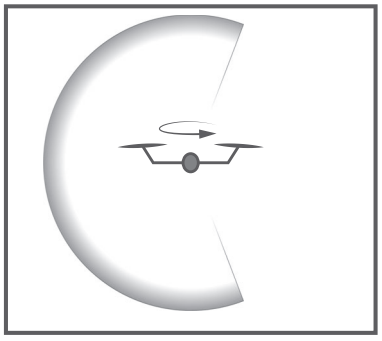
$$\cos(\phi(u,v)) = \frac{f}{\sqrt{(u-u_c)^2 + (v-v_c)^2 + f^2}}. \tag{27}$$

This angle depends on the coordinates of the projected 3D point into the image plane. Once the angle is calculated, the measured depth is divided by the cosine of the angle. This procedure is repeated with all points within the polygon and the results are summed up. To average the value, the resulting sum is divided by the number of points, which were taken for the depth estimation. Equation (28) states the aforementioned calculation

$$d_e = \frac{1}{N_p} \sum_{(u,v)\in P} \frac{d_{pc}(u,v)}{\cos(\phi(u,v))}. \tag{28}$$

Points, which do not have a valid depth estimate are discarded. This might be, because no depth was measured or the point was too close or far away and, thus, discarded for accuracy reasons. The maximum and minimum values for that are parameters of the skill.

5.9 | Exploration turn skill



ARDEA can perform a simple exploration movement to map its environment. Because of the large vertical FOV, rotating around the yaw-axis makes mapping of the nearby surroundings very efficient, rotating roughly 280° is enough to map the whole surrounding of the MAV. In contrast to the Fly-to-waypoint skill from Section 5.2, the

skill is performed on the same spot, but allows rotations greater than 180°, otherwise it shares the same functionality and parameters. A simple waypoint flight cannot provide such a movement, since it always calculates the shortest path between two waypoints, which results in the smallest relative rotation angle that can never be larger than 180°. To monitor whether the desired rotation has been executed, the state machine cannot simply check the current orientation of the MAV, but has to integrate the performed rotation over time. The parameters of the skill are the amount of rotation and the desired rotational velocity. This skill is usually executed with low angular velocity to obtain a high quality map. With this skill the task of mapping an unknown environment can be solved elegantly and time-efficiently.

6 | FIELD EXPERIMENTS

In this section, we demonstrate the capabilities of our MAV ARDEA in field experiments. We show the robustness of the system and apply the presented skill approach to real missions. The modularity of the latter is shown to be useful for defining, monitoring, and performing complex missions.

In the first experiment we illustrate a live demonstration of our flying system on the International Astronautical Congress (IAC). Over five consecutive days, ARDEA performed more than 40 autonomous flights in front of a public audience and experts of the field. In the second experiment, we demonstrate the flight and navigation capabilities of our system on Mount Etna in Italy, which provides a similar textural and geological environment as our moon.

6.1 | IAC mission setup

We performed a public demonstration of our system at the IAC 2018. The IAC is the world's largest annual gathering of space professionals with more than 6,500 participants from 82 different countries, as well as 13,000 visitors on its public open day. At the IAC, we set up an autonomous planetary exploration mission in a Mars-like environment. For that, we used a sandbox with an area of approx. 50 m^2 as shown in Figures 25 and 26. We placed a lander mock-up, as well as large artificial rocks as visual and navigational obstacles for our robots.

Both robots are connected via WLAN to the lander and exchange their submaps as explained in Section 4.2.3. All communication between the ground station and robots was routed through two optical communication terminals, one mounted on the lander (see Figure 25) and the other one a tripod at the opposing side of the sandbox (see Figure 26) to simulate an optical data transmission channel between an orbiter and the lander at the exploration site.

The demonstration features a cooperative exploration mission. ARDEA is placed on the landing platform carried by the LRU. The goal of the robots is to explore a POI which was preselected by the operator. Instead of driving with the LRU directly to the POI and direct its scientific sensor unit on it, the MAV is first sent to investigate the POI since it can reach it faster and already evaluate preliminary if it is advantageous for the rover to drive there. This would be a common procedure in future space missions, where the MAV could also help to find a traversable and fast path for the rover to a POI. After the MAV has investigated the POI, the rover is sent to the location for further investigations. We ran the SLAM framework online and on-board the robots to locally create partial 3D maps,

FIGURE 25 ARDEA and LRU perform an exploration mission at the IAC 2018. One laser terminal is mounted on the top of the lander mock-up in the back [Color figure can be viewed at wileyonlinelibrary.com]

FIGURE 26 Impressions of our cooperative multirobot mapping demonstration at the IAC 2018 with a heterogeneous team of rover and hexacopter: ARDEA observing LRU from its first waypoint after takeoff from the rover's transport platform (top left), ARDEA on its return flight while LRU is navigating to the location explored by ARDEA. One laser terminal is visible on the right side (top right), and ARDEA landed next to its start position while LRU reached its target destination (bottom) [Color figure can be viewed at wileyonlinelibrary.com]

exchange, connect and optimize them as a collaboratively built joint map of the environment.

To execute the mission objectives, skills presented in Section 5 are used as building blocks in a state machine. These are encapsulated into library states in our RAFCON visual state machine programming framework. Figure 27 shows the flow diagram with skills, which was used for the IAC demonstration. The top-down view in Figure 28 illustrates the executed mission sequence.

Using our *modular VO* skill, only VO-0 and VO-1 of the downward looking cameras are activated (see Figures 4 and 17 for numbering of the cameras and VOs), while the upper two VOs are not used for the entire mission. VO-2 was not used because it would have mainly perceived moving persons and therefore given a wrong ego motion estimate. Because the ceiling was above 5 m, the stereo depth estimate is expected to have a high error and hence also VO-3 was not used.

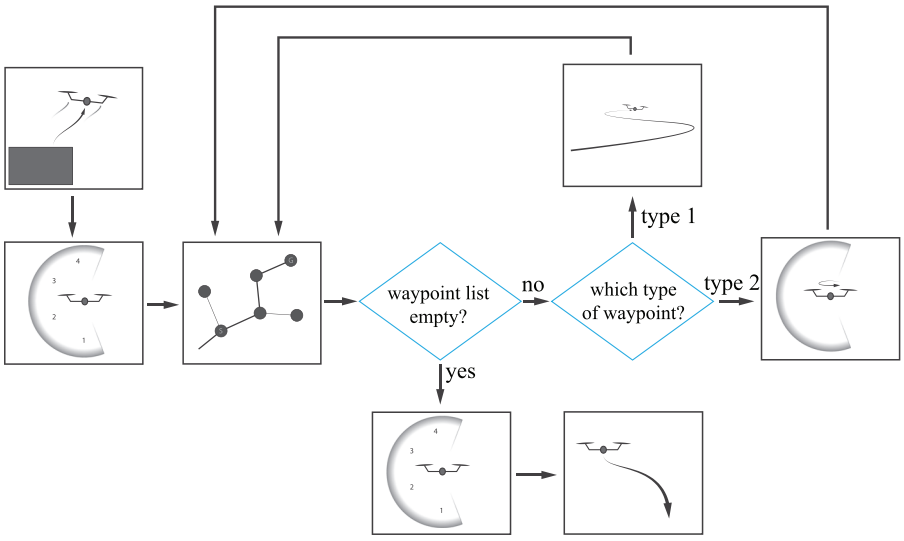


FIGURE 27 Simplified RAFCON state machine which was used for autonomous robot design for extraterrestrial application at the IAC 2018 demo, showing only skills blocks [Color figure can be viewed at wileyonlinelibrary.com]

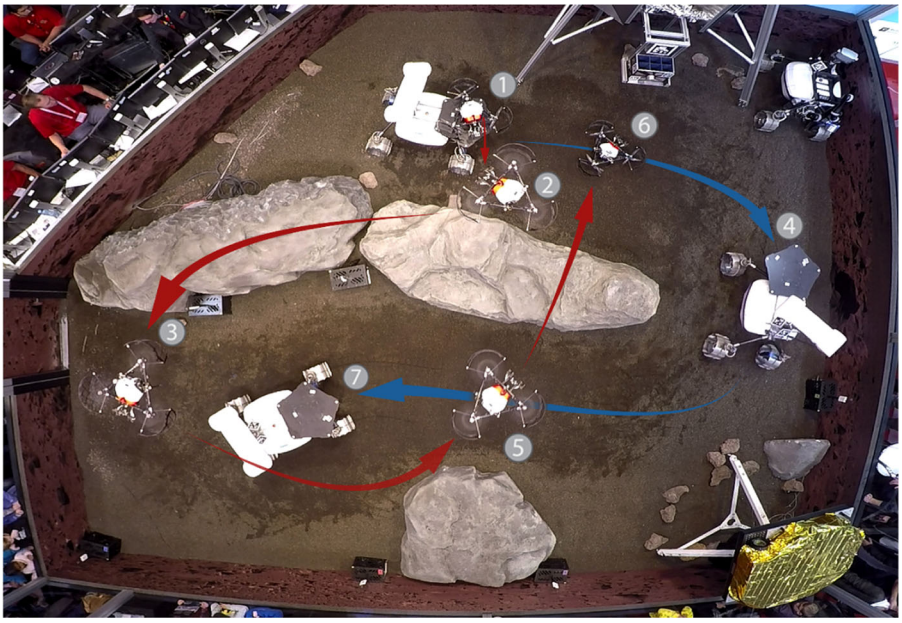


FIGURE 28 Overview of the multirobot collaborative mapping demonstration at the IAC 2018 with LRU (blue) and ARDEA (red): ARDEA takes off from LRUs transport platform, then at its first waypoint, it turns back and detects LRU to connect their pose and map estimates, then performs a more than 360° scan at the POI [Color figure can be viewed at wileyonlinelibrary.com]

When the mission operator is ready, ARDEA waits for the start command to execute the *Takeoff* skill (see Section 5.1) ① and start investigating the previously set POI. After ARDEA took off from LRU, VO-1 gets deactivated for the same reason as VO-2. Only VO-0 remains active, it is expected to give the best state estimate from this point on. This highly configurable approach shows how the mission operator can modify the VO setup to deal with challenges found in the operation environment. Another challenge was posed by the small granules on the ground, which were partially blown away due to the downwash of the system's propulsion system. Since the particles are flying away in a structured manner, they could have caused false ego motion estimations. However VO-0 was able to filter out these outliers and track enough static environment features to not get fooled by the moving particles.

After takeoff ARDEA queries the path from start to goal waypoint in the *Waypoint Planning* skill (see Section 5.7). Once the MAV receives the next waypoint, it calculates its relative pose to it and sends it as input to the low-level control skill *Fly to Waypoint*. The current pose estimate is compared to the current desired waypoint at a rate of 2.5 Hz and a error margin of 30 cm to determine whether the current waypoint has been reached and then requests the next waypoint in the queried list.

In the next step ② the MAV rotates around to detect the rover, calculates the relative transformation between both vehicles and uses it to align both local maps which are shared over the wireless communication channel. For this, several AprilTags are used as fiducial markers (Olson, 2011) on the rover. ARDEA is then flying to the POI ③. Upon arrival it performs the *Exploration turn* skill to quickly explore the area. Once this information is processed and transferred to the ground-station, the human operator can pick a POI and sends LRU there for further investigation ④. In the case of this demo, which was repeated around 10 times per day, this POI was preprogrammed. Finally the MAV has performed its task and hence returns to the lander site. Instead of flying back directly, it takes a short detour to map more of its immediate environment ⑤. When the lander was reached, VO-1 is activated again to aid VO-0 during the landing maneuver, where the ground might be already too close to get robust feature detections and depth estimates. While the *Landing* skill is executed ⑥, the LRU is driving autonomously to the POI ⑦. Figure 29 illustrates the 3D voxel map and pointcloud jointly created by both systems in a time series and Figure 30 shows the final jointly crated RGB pointcloud.

6.2 | ROBEX mission

To test our system in a moon analogue scenario, we performed a variety of experiments on Mt. Etna, Italy as part of the Robotic Exploration of Extreme Environments (ROBEX) mission[7]. A detailed study of the lunar crust's structure was the main scientific objective in ROBEX. The two main missions were to install a seismic network and to perform a seismic profile measurement. A number of other experiments aimed to demonstrate the capabilities in autonomous navigation, multirobot mapping, crater exploration, geological sample investigation, and probe placement (Wedler et al., 2017, 2018). Its natural seismic activity and volcanic origin, as well as its representativeness of lunar surfaces with respect to topography and morphology fulfilled the criteria of Moon-analogousness. In addition, its similarity in terms of geological context and shape was important to test the visual navigation components. Our test site was located next to the *Cisternazza*

[7]ROBEX demo-missions: http://www.robex-allianz.de/en/about-robex/demo-missions

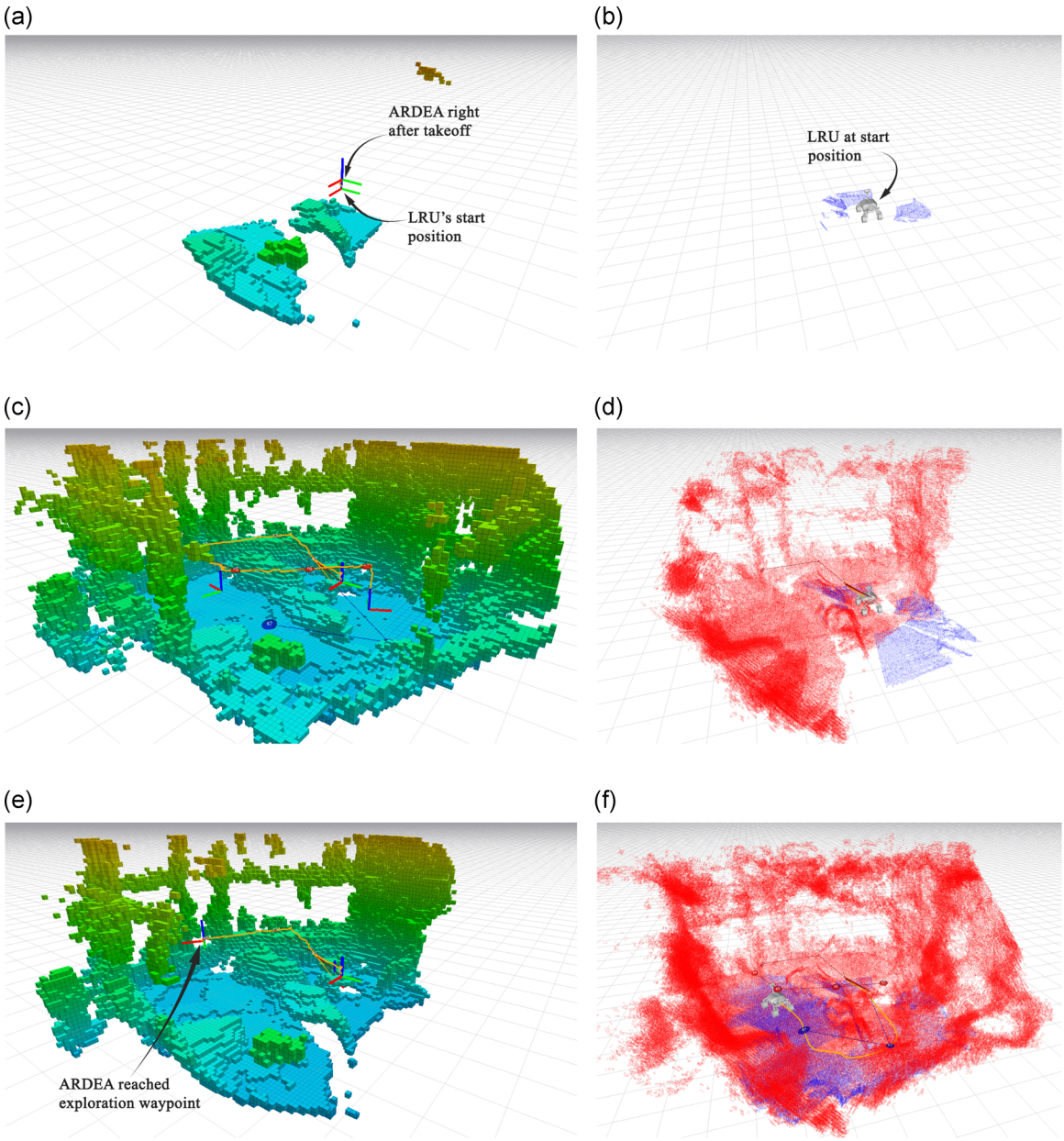


FIGURE 29 Image sequence showing the multirobot mapping by LRU and ARDEA during one of our heterogeneous multirobot experiments at the IAC 2018 as computed on-board ARDEA. Left: height-colored probabilistic voxel grid map. Right: robot-colored pointcloud map, blue: LRU, red: ARDEA. (a) t = 1 min 27 s; (b) t = 1 min 27 s; (c) t = 2 min 09 s; (d) t = 2 min 09 s; (e) t = 3 min 45 s; (f) t = 3 min 45 s [Color figure can be viewed at wileyonlinelibrary.com]

crater at an altitude of about 2,600 m above sea-level. Testing our hardware and autonomy software setup there was challenging for a variety of reasons. At this altitude wind strength and direction were fluctuating frequently, causing the attitude controller to operate close to its attitude limits. Another difficulty was the volcanic soil which has magnetic and electric conductive properties, it can therefore, harm any robotic system by causing short-circuits on PCBs as a worst case scenario. Moreover, fine grained swirled up soil particles from downwash can creep into the motors and hence cause additional friction and resulting heat or even block the motors completely. Eventually, moving particles can lead to wrong ego motion estimates in the VO, as also discussed in the IAC experiment (see Section 6.1).

Although many issues arise from flying low over the ground, it is still the best strategy to explore and map the environment because the map and pose estimation quality deteriorate with increasing camera to ground distance. In the following, we show a test flight performed on Mt. Etna, which is showing that our visual-inertial navigation and mapping framework still performs well even under the harsh conditions mentioned above.

The upper image of Figure 31 shows the black and white converted image of the left pinhole cameras. Notice that we are just

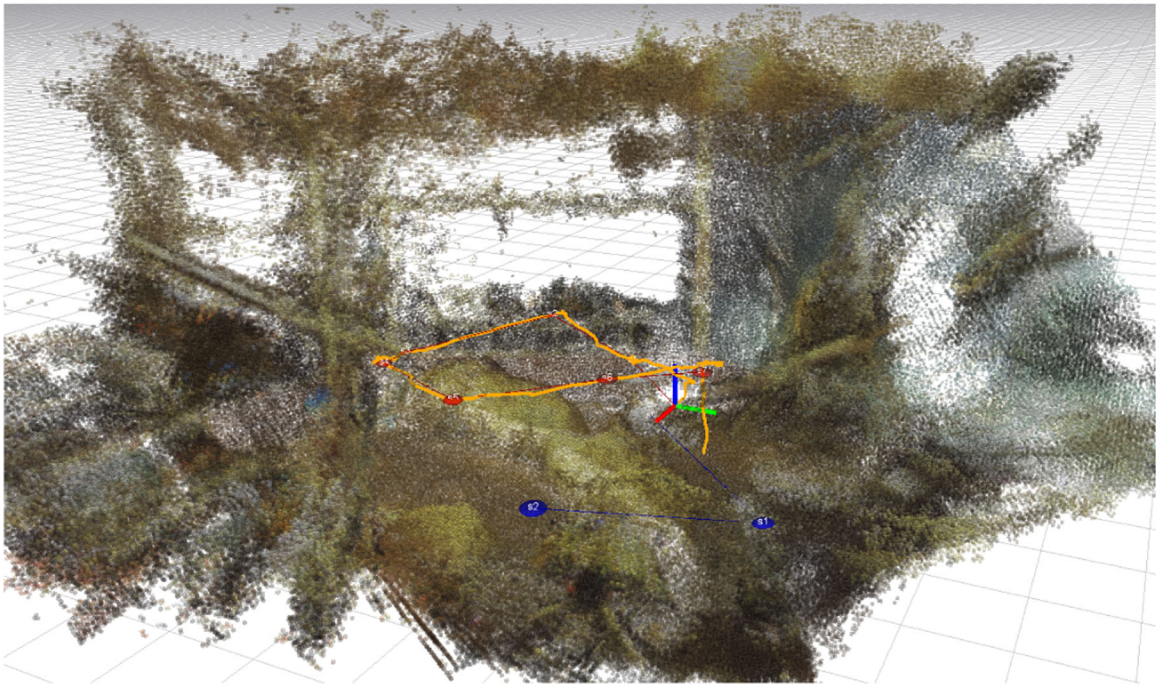


FIGURE 30 Final RGB pointcloud generated by ARDEA and LRU at the IAC 2018 demo. The orange line is illustrating the flown path and the coordinate system marks the start position of ARDEA [Color figure can be viewed at wileyonlinelibrary.com]

using images of three pinhole cameras for this experiment (explanation see Figure 5). Since we are performing our test on an open field, the uppermost virtual pinhole cameras 3 L and 3R would just show the bright sky and therefore not usable for navigation. Even the virtual pinhole cameras 2L and 2R show just the sky most of time. However, depending on the flight trajectory, it sometimes also captures the surrounding at higher elevations. Although this far distance camera view is not useful for ego estimation in the VO, it can still be used for other mission-relevant tasks. The VO of pinhole camera 1 covers a good range of far distance and close-up features, which makes it robust against fast movements and still accurate during slow maneuvers. Therefore VO-1 is used as main VO, but depending on the situation, the VO setup can be altered.

The lower image of Figure 31 illustrates the corresponding depth image. Note that the depth is dense in the fore- and mid-ground. The depth for the box, far away hills, and the sky is invalid. The box is too close to the camera and the hills are too far away to get a sufficiently good depth estimate. The sky, on the other hand, does not provide any useful gradients and as a result no good features. However, even if clouds were present, they would be too far away to perform a valid depth estimate.

The upper left image of Figure 31 shows the tracked features of VO-1. It picks up most features in the close range, since the gradients are greater and the depth estimation better compared to the other image regions.

Figure 32 shows the estimated trajectory during flight. Due to time and weight limitations, no GNSS-based ground truth was available. But it is known that after 110 s flight time, the take-off and landing place were approximately 0.3 m apart. This results in an approximate error in the lower single-digit percentage range with respect to the accumulated trajectory length.

7 | DISCUSSION AND CONCLUSION

In this paper, we presented ARDEA, an MAV equipped with skills targeted for future planetary missions. Its skill primitives and state-of-the-art vision sensor based setup makes it suitable for indoor as well as outdoor environments and hence various application scenarios. Due to the full on-board perception and mapping of the environment, the MAV can act completely autonomous and does not rely on a communication channel during a mission. Therefore, the MAV can navigate through cluttered environments and has proven to cope well in semidynamic scenes with moving particles on the floor, which was demonstrated in two-field experiments. Compared to previous work (Schmid, Lutz, et al., 2014; Tomić et al., 2012) the presented multirotor platform is completely customized, it has a redundant propeller configuration and a ultra-wide fisheye camera FOV which covers ground and ceiling in indoor environments. Moreover, the fully accessible ESCs enable sophisticated custom control algorithms such as the presented external wrench estimation to observe and explicitly model external disturbances which adds additional robustness to the system. Although ARDEA was primarily designed for planetary missions, it is versatile enough to be used for various missions, for example, also for terrestrial applications. For example, in Touko Tcheumadjeu et al. (2018), we have shown a commercial

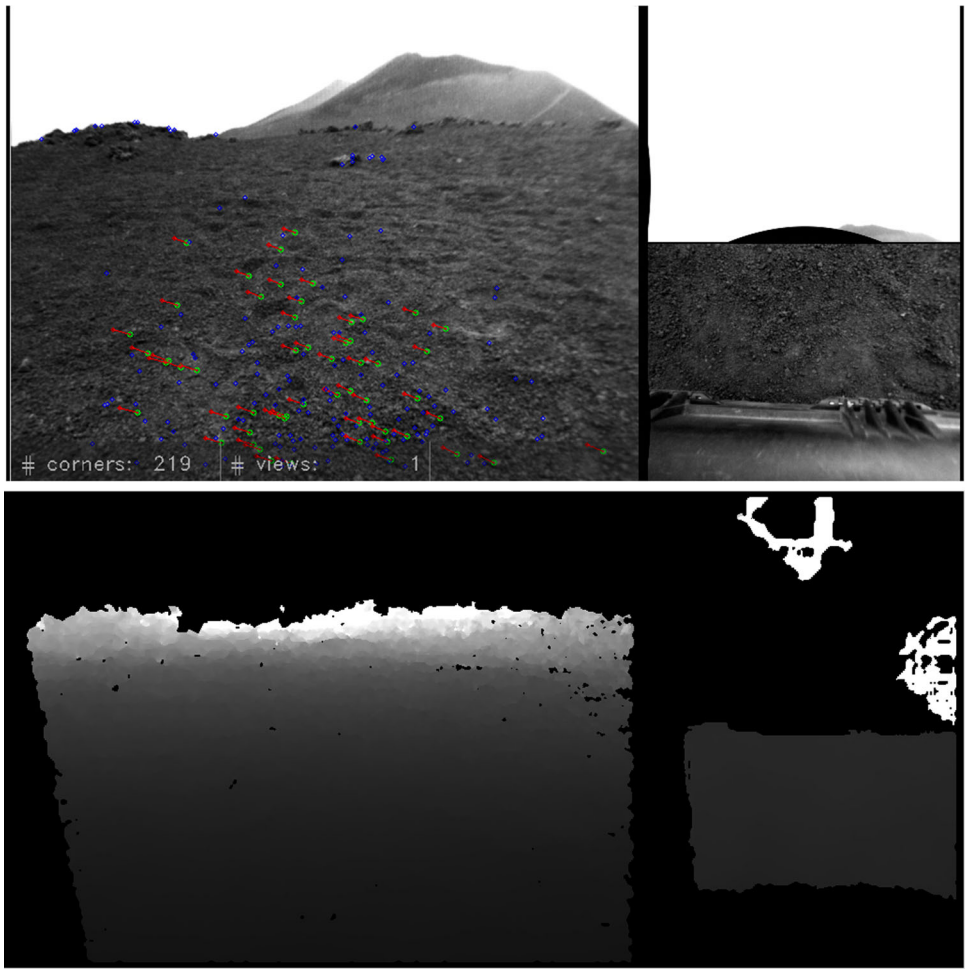


FIGURE 31 Top: Images of three cameras, left: forward pointing camera with tracked corners, top-right: upward pointing camera, bottom-right: downward pointing camera. Bottom: Resulting depth image of upper input images [Color figure can be viewed at wileyonlinelibrary.com]

application within the EC-funded H2020 project Automated driving Progressed by Internet Of Things (AUTOPILOT), in which ARDEA was used to locate free parking spots to guide autonomous cars. Moreover, the navigation software is not restricted to flying vehicles and can be used on ground-based mobile robots. In the following sections, we discuss the lessons that we learned while building the presented system. Finally, we discuss possible future work, addressing remaining open challenges for planetary exploration scenarios.

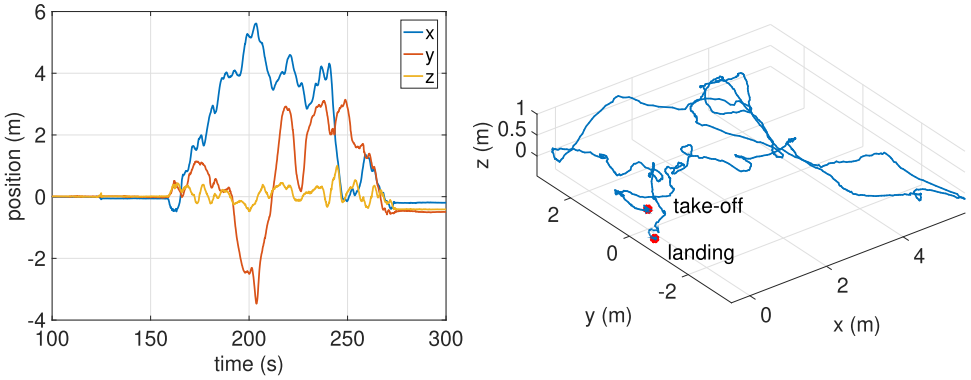


FIGURE 32 Trajectory of flight on Mt. Etna, left: estimated *x*, *y*, and *z* position with respect to take-off position over time, right: three-dimensional flight trajectory [Color figure can be viewed at wileyonlinelibrary.com]

7.1 | Lessons learned

During the design and operation of ARDEA we learned a few lessons which we would like to share with the community. This covers experiences with the general hardware design, sensor choice, and calibration as well as the autonomy software components.

7.1.1 | Hardware design and sensor choice

Separating the MAV into a navigation stack and a propulsion system (frame) makes it a modular platform that allows to easily exchange sensor stacks or frames. For example, during manual flight testing, we usually work with a slimmer navigation stack without any cameras attached. This has the advantage that we do not damage our camera sensors in case of a crash during risky control tests involving experimental control algorithms. However even in case of a crash, most energy is absorbed by the frame, while the impact on the stack electronics is reduced by the rubber dampers between stack and frame. During failed experiments involving a crash, this saved us time and money for replacing components on the stack. In addition, the dampers, together with the combined weight of all stack components, function as a mechanical low-pass filter for vibration damping, see design concept in Section 3.1.3. In the past, we had seen unstable control responses due to positive feedback loops caused by vibrations while we were not using vibration damping. We also learned that it is valuable that the navigation stack can easily be attached and detached without decalibrating the stereo camera configuration or changing the camera-to-IMU transformations.

Finally, the navigation stack can be attached to other mobile robots or used as a hand-held device, which makes it a reusable self-contained unit for camera-based navigation and mapping, requiring only a power supply as input and providing a CAN-bus interface to control actuators.

Using a wide-angle FOV stereo camera system has shown to increase the level of robustness in two ways. First, it improves the VO ego-motion estimation (Section 4.2.1) by being able to use structures on the ground as well as the ceiling in indoor scenarios. Second, it detects obstacles in a wide range, which makes it useful for selecting safe landing sites and allows the trajectory planner a greater movement planning space in which obstacles are observable. On our previous system (Schmid, Lutz, et al., 2014), we did not have this wide FOV and once encountered a crash due to a structure hanging down from the ceiling as it was not observable by the narrow FOV stereo cameras and hence not considered as an obstacle by the path planner. A wide FOV also makes mapping more efficient: With our system, a turn of less than 300° is enough to get a 3D map of the whole surrounding environment. However, using four cameras with a large FOV comes with the price of a difficult camera calibration procedure. For calibration of intrinsic and extrinsic parameters, we are using a classical checkerboard. While taking calibration images, it is important that the pattern frequently appears in the lower and upper cameras. To get a good extrinsic calibration, images of the pattern should also be taken in the complete camera field of view which is 240° vertically and 80° horizontally, this is not always easy to accomplish and requires a big checkerboard.

7.1.2 | Autonomy software architecture

Due to the fact that MAVs are inherently unstable, it is essential that the controller software runs robustly. One way is to outsource this task to a dedicated microcontroller in contrast to a computer with an operating system that is running several tasks in parallel. This however comes with the disadvantage of high debugging complexity, heterogeneous software development tools and APIs. On the proposed flying platform we followed the established approach of separating the autonomy software into high level and low level RT tasks and distributing them on different computers so that they can influence each other only by predefined middleware interfaces. This separation improves the robustness of the system because high level tasks, such as navigation and mapping, require a high data throughput and processing time do not interfere with low level high frequency tasks, such as the attitude and position controller. Moreover, dynamic memory (re)allocations common in computer vision software might interfere with the real-time scheduling and can even fail due to running out of memory. In our approach however, unlike other microcontroller driven platforms such as PX4[8], we use a RT Linux OS for deployment of attitude and position controllers and proved stable operation across both presented field experiments, see Section 6.

Using visual-inertial state estimation with multiple odometries has proven to be helpful in the IAC mission, see Section 6.1. With this setup it was easy to tailor the navigation system to a specific mission and therefore fly in difficult environments with moving particles and structures. Of course, as in M. G. Müller et al. (2018), all VOs could be activated all the time and the local reference filter would fuse different VO measurements based on their covariance estimates. However, in deterministic situations the human operator has a better understanding of the particular environment or mission challenges and can choose which VO to use manually for robust operation.

The skill setup has shown its benefits in the experiments and development phase of the system. Also, if skills are defined generically enough, they can also be re-used across other mobile robots. By using skills, a human operator has intuitive access to the robot's capabilities and can assemble complex missions in a quick manner, without the need to have a fundamental understanding of the underlying algorithms. Those missions are broken down into a sequence of a few high level skills, which in turn contain lower level skills. This hierarchical breakdown of skills in our state machine framework RAFCON helped us a lot in visualizing and debugging state transitions and component failures in real-time. Already early on during the

[8] https://px4.io

development phase we could quickly locate bugs and problems in software modules using this approach.

7.2 | Future work

To be able to use ARDEA for future planetary missions, open challenges regarding hardware and autonomy software still have to be solved. This section summarizes work in progress and visionary ideas for accomplishing such a mission.

7.2.1 | Space qualified hardware

A major step for raising the technology readiness level (TRL) towards a final space design is re-designing the system with space qualified hardware components which can handle high radiation and temperature levels. Another problem is the rather low computational power of current state-of-the-art space qualified computers compared to consumer grade computers. Using only cameras as main exteroceptive sensor has the advantage of straightforward space qualification, unlike current mechanically driven LIDARs. However, solid-state LIDARs without moving parts seem to be a promising alternative in the future. Although increasing the TRL is an important step towards a real space mission, we have been mainly focusing on design concepts and algorithms using terrestrial hardware. This allows for fast development iterations and proof of research concepts.

7.2.2 | Skills based on semantic environment interpretation

Because a high level of autonomy can make planetary exploration missions safer and more efficient, we are planning to add skills based on semantic interpretation of camera images to our system. This can be used for terrain classification or the detection of known and unknown structures, that means it can help to autonomously identify scientifically interesting geological features such as soils or rock without constant supervision by planetary science researchers. Due to long round-trip times or even offline phases in some missions, constant supervision is not even possible. Those algorithms are commonly referred to as novelty detection (Pimentel, Clifton, Clifton, & Tarassenko, 2014) within the statistics and machine learning community. Moreover, semantic image segmentation can also help to improve the accuracy of the VO and mapping algorithms in scenarios with dynamic environments, for example by distinguishing between static and dynamic objects and therefore disregarding dynamic objects for ego-motion estimation.

Since the MAV is limited in computational power mainly due to weight constraints, it opens up challenging research questions on how to obtain and process semantically labeled data. During a terrain classification scenario, images are acquired and a novelty detection algorithm detects and clusters geological features. In an offline phase, the classification results can be assessed by the operation researchers and custom labels assigned to interesting features. To robustly recognize the labeled features in new data, an incremental learning approach (Bruzzone & Prieto, 1999) can be used. Because the on-board computers are running near their load limit during flight, training phases can only be done while the MAV is not flying. In missions without the option to outsource offline computations to other robots or the lander, dedicated *sleep phases* of flight inactivity might be a solution to realize the learning steps. The inference step, however, has to be implemented efficiently enough to run online on the MAV during the mission.

Another important new functionality would be the skill to land on complex terrain like boulders and structured objects like another robot. Semantic interpretation of the ground can help to understand the difficulty to land on certain terrains and which landing strategy to use accordingly.

For the mapping framework we plan to match submaps created by ARDEA to gain further intra- and inter-robot loop closure constraints for relocalization based on the 3D geometry of the environment, indicated by the dashed line in Figure 21. We already implemented and demonstrated such a method for teams of planetary exploration rover prototypes in Schuster et al. (2018). In future work, we could also reduce this computational effort by restricting the compositions to application-dependent requests and limiting them to regions of interest. An example would be a path planning algorithm requesting the computation of a global map estimate of areas that are to be traversed next as an active navigation approach.

To be able to operate ARDEA even more intuitively in the future, it would be interesting to deploy innovative human machine interfaces. Since our MAV provides a wide FOV, methods and hardware developed for the VR and AR community might be suitable.

7.2.3 | Robot teams

Most MAVs still have limited computational power and flight time as well as a lack of manipulating capabilities and therefore can just be used for scouting purposes. To solve these problems a team of heterogeneous robots can be utilized to solve complex missions (Wedler et al., 2018). One team member might be used for scouting, like an MAV, whereas another member of the team is especially designed for manipulation tasks. The MAV, for instance, can search for interesting objects and communicate their positions to the other team member that can then pick them up for further analysis. A more computational powerful robotic team member can also help to tackle the problem of limited computational resources of MAVs by taking over noncritical computations, which might also result in increased flight time. Such an example could be the scientific evaluation of captured image data or the computationally intensive learning phase in the terrain classification/novelty detection use case. In all of these cases the here presented skill setup is of great help, since it can be used across robotic team members. Therefore, in the future we want to further focus on developing a team of robots, which can share skills and tasks.

ACKNOWLEDGMENTS

This study was supported by the Helmholtz Association, project alliance ROBEX (contract number HA-304), the project ARCHES (contract number ZT-0033) and the EU-project AUTOPILOT (contract number 731993). We want to thank the RMC workshops and system administration for their continuous support, Andreas Dömel for fruitful discussions about robotic skills and Jongseok Lee for contributing deep learning software to the system. Special thanks goes to Harald Wagner for manufacturing mechanical parts and Benedikt Pleintinger, Rene Wagner, Nikolaus Seitz and Simon Schätzle for support with the on-board electronics.

ORCID

Philipp Lutz http://orcid.org/0000-0002-4552-8509
Martin J. Schuster http://orcid.org/0000-0002-6983-3719

REFERENCES

Achtelik, M., Achtelik, M., Weiss, S., & Siegwart, R. (2011). Onboard imu and monocular vision based control for MAVs in unknown in- and outdoor environments. *2011 IEEE International Conference on Robotics and Automation* (pp. 3056–3063).

Achtelik, M. W., Lynen, S., Chli, M., & Siegwart, R. (2013). Inversion based direct position control and trajectory following for micro aerial vehicles. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2933–2939).

Augugliaro, F., Lupashin, S., Hamer, M., Male, C., Hehn, M., Mueller, M. W., & D'Andrea, R. (2014). The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems Magazine*, *34*(4), 46–64.

Balaram, B., Canham, T., Duncan, C., Grip, F., Johnson, H., Maki, W., & Zhu, D. (2018). Mars helicopter technology demonstrator. *AIAA Atmospheric Flight Mechanics Conference*.

Barry, A. J., & Tedrake, R. (2015). Pushbroom stereo for high-speed navigation in cluttered environments. *IEEE International Conference on Robotics and Automation* (pp. 3046–3052).

Beul, M., Krombach, N., Nieuwenhuisen, M., Droeschel, D., & Behnke, S. (2017). Autonomous navigation in a warehouse with a cognitive micro aerial vehicle. In A. Koubaa (Ed.), *Robot operating system (ROS): The complete reference (Volume 2)* (pp. 487–524). Cham: Springer International Publishing.

Beul, M., Krombach, N., Zhong, Y., Droeschel, D., Nieuwenhuisen, M., & Behnke, S. (2015). A high-performance mav for autonomous navigation in complex 3d environments. *2015 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 1241–1250).

Bloesch, M., Omari, S., Hutter, M., & Siegwart, R. (2015). Robust visual inertial odometry using a direct ekf-based approach. ETH-Zrich. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Conference Location: Hamburg, Germany; Conference Date: September 28–October 2, 2015.

Brand, C., Schuster, M. J., Hirschmüller, H., & Suppa, M. (2015). Submap matching for stereo-vision based indoor/outdoor SLAM. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Brunner, S. G., Steinmetz, F., Belder, R., & Dömel, A. (2015). RAFCON: A graphical tool for task programming and mission control. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Bruzzone, L., & Prieto, D. F. (1999). An incremental-learning neural network for the classification of remote-sensing images. *Pattern Recognition Letters*, *20*(11), 1241–1248.

Chartrand, R., & Yin, W. (2008). Iteratively reweighted algorithms for compressive sensing. *IEEE International Conference on Acoustics, speech and signal processing, 2008. ICASSP 2008*. IEEE (pp. 3869–3872).

Daga, A. W., Allen, C., Battler, M. M., Burke, J. D., Crawford, I. A., Léveillé, R. J., & Tan, L. T. (2009). Lunar and Martian Lava Tube exploration as part of an overall scientific survey. *Annual Meeting of the Lunar Exploration Analysis Group*, volume 1515 of *LPI Contributions*, Vol. 15.

De Boor, C. (2001). *A practical guide to splines*. New York, NY: Springer.

Dellaert, F. (2015). GTSAM 3.2.1. Retrieved from https://collab.cc.gatech.edu/borg/gtsam

Delmerico, J., & Scaramuzza, D. (2018). A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271.

Engel, J., Koltun, V., & Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*, 611–625.

Faessler, M., Franchi, A., & Scaramuzza, D. (2018). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, *3*(2), 620–626.

Falanga, D., Kleber, K., Mintchev, S., Floreano, D., & Scaramuzza, D. (2019). The foldable drone: A morphing quadrotor that can squeeze and fly. *IEEE Robotics and Automation Letters*, *4*, 209–216.

Forster, C., Carlone, L., Dellaert, F., & Scaramuzza, D. (2017). On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, *33*(1), 1–21.

Forster, C., Zhang, Z., Gassner, M., Werlberger, M., & Scaramuzza, D. (2017). SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*.

Gohl, P., Honegger, D., Omari, S., Achtelik, M., & Siegwart, R. (2015). Omnidirectional visual obstacle detection using embedded FPGA. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Guennebaud, G. (2017). Quadprog: C++ solver for quadratic programming problems. Retrieved from https://github.com/ggael/QuadProg

Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. Retrieved from http://eigen.tuxfamily.org

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107.

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

Hirschmüller, H., Innocent, P. R., & Garibaldi, J. M. (2002). Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. *7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002* (pp. 1099–1104).

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, *34*(3), 189–206.

Houben, S., Quenzel, J., Krombach, N., & Behnke, S. (2016). Efficient multi-camera visual-inertial SLAM for micro aerial vehicles. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Huber, M. (2016). NASA Helicopter Could Fly on Mars.

ISECG. (2018). *Global exploration roadmap* (Technical report). Author.

Johnson, S. G. (2008). The NLopt non-linear optimization package. Retrieved from http://ab-initio.mit.edu/nlopt

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research*, *31*, 217–236.

Kamel, M. S., Verling, S., Elkhatib, O., Sprecher, C., Wulkop, P., JeremyTaylor, Z., & Gilitschenski, I. (2018). The Voliro omnidirectional hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle. *IEEE Robotics & Automation Magazine*, *25*, 34–44.

Kang, C., Fahimi, F., Griffin, R., Landrum, D. B., Mesmer, B., Zhang, G., & McCain, J. (2019). *Marsbee—swarm of flapping wing flyers for enhanced mars exploration. phase i: Final report* (Techreport HQ-E-DAA-TN67472), NASA.

Kraft, D. (1988). A software package for sequential quadratic programming. DFVLR, Institut fr Dynamik der Flugsysteme, Oberpfaffenhofen, Germany.

Kuhl, C. (2008). Design of a mars airplane propulsion system for the aerial regional-scale environmental survey (ares) mission concept. *44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*.

LaValle, S. M., & Kuffner, J J., Jr. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, *20*(5), 378–400.

Lee, T., Leoky, M., & McClamroch, N. H. (2010). Geometric tracking control of a quadrotor UAV on se(3). *49th IEEE Conference on Decision and Control (CDC)* (pp. 5420–5425).

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., & Furgale, P. (2015). Keyframe-based visualinertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, *34*(3), 314–334.

Liu, S., Mohta, K., Atanasov, N., & Kumar, V. (2018). Search-based motion planning for aggressive flight in se(3). *IEEE Robotics and Automation Letters*, *3*(3), 2439–2446.

Lorenz, R. D., Turtle, E. P., Barnes, J. W., Trainer, M. G., Adams, D. S., Hibbard, K. E., & Bedini, P. D. (2017). *Dragonfly: A rotorcraft lander concept for scientific exploration at Titan* (Technical report). Johns Hopkins APL Technical Digest.

Mahony, R., Kumar, V., & Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics Automation Magazine*, *19*(3), 20–32.

Maimone, M., Johnson, A., Cheng, Y., Willson, R., & Matthies, L. (2006). Autonomous navigation results from the mars exploration rover (mer) mission. In M. H. Ang & O. Khatib (Eds.), *Experimental Robotics IX* (pp. 3–13). Berlin, Heidelberg: Springer.

Mair, E., Hager, G. D., Burschka, D., Suppa, M., & Hirzinger, G. (2010). Adaptive and generic corner detection based on the accelerated segment test. *Proceedings of the 11th European Conference on Computer Vision: Part II*, ECCV'10, Berlin, Heidelberg: Springer-Verlag (pp. 183–196).

Maki, J., Thiessen, D., Pourangi, A., Kobzeff, P., Litwin, T., Scherr, L., & Maimone, M. (2012). The mars science laboratory engineering cameras. *Space Science Reviews*, 170.

Mankins, J. (1987). *Mars rover technology workshop proceedings*. Technical report, Jet Propulsion Laboratory (JPL) Internal Report D-4788, Pasadena, CA.

Matthies, L., Brockers, R., Kuwata, Y., & Weiss, S. (2014). Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. *IEEE International Conference on Robotics and Automation (ICRA)*.

Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. *2011 IEEE International Conference on Robotics and Automation* (pp. 2520–2525).

Michieletto, G., Ryll, M., & Franchi, A. (2018). Fundamental actuation properties of multirotors: Force-moment decoupling and fail-safe robustness. *IEEE Transactions on Robotics*, *34*(3), 702–715.

Mourikis, A., & Roumeliotis, S. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. *2007 IEEE International Conference on Robotics and Automation* (pp. 3565–3572).

Müller, M., & D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 45–52).

Müller, M. G., Steidle, F., Schuster, M. J., Lutz, P., Maier, M., Stoneman, S., & Stürzl, W. (2018). Robust visual-inertial state estimation with multiple odometries and efficient mapping on an MAV with ultra-wide FOV stereo vision. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Müller, M. W., & D'Andrea, R. (2016). Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles. *The International Journal of Robotics Research*, *35*(8), 873–889.

Müller, M. W., Hehn, M., & D'Andrea, R. (2015). A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Transactions on Robotics*, *31*(6), 1294–1310.

Mur-Artal, R., & Tardós, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*.

Nägeli, T., Alonso-Mora, J., Domahidi, A., Rus, D., & Hilliges, O. (2017). Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, *2*(3), 1696–1703.

Nieuwenhuisen, M., & Behnke, S. (2015). 3D planning and trajectory optimization for real-time generation of smooth MAV trajectories. *2015 European Conference on Mobile Robots (ECMR)* (pp. 1–7).

Ogasawara, T., Kitagaki, K., Suehiro, T., Hasegawa, T., & Takase, K. (1993). Model based implementation of a manipulation system with artificial skills. In R. Chatila & G. Hirzinger (Eds.), *Experimental Robotics II* (pp. 89–98). Berlin, Heidelberg: Springer.

Ok, K., Gamage, D., Drummond, T., Dellaert, F., & Roy, N. (2015). Monocular image space tracking on a computationally limited MAV. *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6415–6422).

Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., & Galceran, E. (2016). Continuous-time trajectory optimization for online UAV replanning. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5332–5339).

Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. *IEEE International Conference on Robotics and Automation (ICRA)*.

Oskiper, T., Zhu, Z., Samarasekera, S., & Kumar, R. (2007). Visual odometry system using multiple stereo cameras and inertial measurement unit. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Peters, J., Kober, J., Mülling, K., Nguyen-Tuong, D., & Kroemer, O. (2012). Robot skill learning. *20th European Conference on Artificial Intelligence* (pp. 40–45).

Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, *99*, 215–249.

Qin, T., Li, P., & Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, *34*(4), 1004–1020.

Richter, C., Bry, A., & Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments (pp. 649–666). Cham: Springer International Publishing.

Schauwecker, K., & Zell, A. (2014). On-board dual-stereo-vision for the navigation of an autonomous MAV. *Journal of Intelligent & Robotic Systems*, *74*(1), 1–16.

Schmid, K., Lutz, P., Tomić, T., Mair, E., & Hirschmüller, H. (2014). Autonomous vision-based micro air vehicle for indoor and outdoor navigation. *Journal of Field Robotics*, *31*(4), 537–570.

Schmid, K., Ruess, F., & Burschka, D. (2014). Local reference filter for life-long vision aided inertial navigation. *17th International Conference on Information Fusion*.

Schmid, K., Ruess, F., Suppa, M., & Burschka, D. (2012). State estimation for highly dynamic flying systems using key frame odometry with varying time delays. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Schneider, J., & Förstner, W. (2015). Real-time accurate geo-localization of a MAV with omnidirectional visual odometry and gps, *Computer Vision—ECCV 2014 Workshops* (pp. 271–282). Cham: Springer International Publishing.

Schuster, M. J., Brand, C., Hirschmüller, H., Suppa, M., & Beetz, M. (2015). Multi-robot 6D graph SLAM connecting decoupled local reference filters. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Schuster, M. J., Brunner, S. G., Bussmann, K., Büttner, S., Dömel, A., Hellerer, M., & Armin, W. (2017). Towards autonomous planetary exploration. *Journal of Intelligent & Robotic Systems*.

Schuster, M. J., Schmid, K., Brand, C., & Beetz, M. (2018). Distributed stereo vision-based 6D localization and mapping for multi-robot teams. *Journal of Field Robotics*.

Smeur, E. J. J., de Croon, G. C. H. E., & Chu, Q. (2017). Cascaded incremental nonlinear dynamic inversion control for MAV disturbance rejection. *Control Engineering Practice*.

Steinmetz, F., & Weitschat, R. (2016). Skill parametrization approaches and skill architecture for human-robot interaction. *2016 IEEE International Conference on Automation Science and Engineering (CASE)* (pp. 280–285).

Stelzer, A., Hirschmüller, H., & Görner, M. (2012). Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain. *The International Journal of Robotics Research*, *31*, 381–402.

Stoneman, S., & Lampariello, R. (2016). A nonlinear optimization method to provide real-time feasible reference trajectories to approach a tumbling target satellite. *13th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Vol. 13.

Sussman, G. J. (1973). A computational model of skill acquisition. PhD thesis, Massachusetts Institute of Technology.

Taylor, J. R. (1982). *An introduction to error analysis*. Sausalito, CA: University Science Books.

Thangavelautham, J., Robinson, M. S., Taits, A., McKinney, T., Amidan, S., & Polak, A. (2014). Flying, hopping Pit-Bots for cave and lava tube exploration on the Moon and Mars. *2nd International Workshop on Instrumentation for Planetary Missions*.

Tomić, T., Lutz, P., Schmid, K., Mathers, A., & Haddadin, S. (2018). Simultaneous contact and aerodynamic force estimation (s-cafe) for aerial robots. *International Journal of Robotics Research (IJRR)*.

Tomić, T., Ott, C., & Haddadin, S. (2017). External wrench estimation, collision detection, and reflex reaction for flying robots. *IEEE Transactions on Robotics*, *33*(6), 1467–1482.

Tomić, T., Schmid, K., Lutz, P., Dömel, A., Kassecker, M., Mair, E., & Burschka, D. (2012). Toward a fully autonomous UAV. *IEEE Robotics & Automation Magazine*, *19*, 46–56.

Tomić, T., Schmid, K., Lutz, P., Mathers, A., & Haddadin, S. (2016). The flying anemometer: Unified estimation of wind velocity from aerodynamic power and wrenches. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1637–1644).

Touko Tcheumadjeu, L. C., Andert, F., Tang, Q., Sohr, A., Kaul, R., Belz, J., & Stürzl, W. (2018). Integration of an automated valet parking service into an internet of things platform. *IEEE International Conference on Intelligent Transportation Systems (ITSC)*.

Uszenko, V., vonStumberg, L., Pangercic, A., & Cremers, D. (2017). Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 215–222).

Wang, R., Schwörer, M., & Cremers, D. (2017). Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. *International Conference on Computer Vision (ICCV)*, Venice, Italy.

Wedler, A., Vayugundla, M., Lehner, H., Lehner, P., Schuster, M., Brunner, S., & Wilde, M. (2017). First results of the robex analogue mission campaign: Robotic deployment of seismic networks for future lunar missions. *68th International Astronautical Congress (IAC)*.

Wedler, A., Wilde, M., Dömel, A., Müller, M. G., Reill, J., Schuster, M., & Albu-Schäffer, A. O. (2018). From single autonomous robots toward cooperative robotic interactions for future planetary exploration missions. *69th International Astronautical Congress (IAC)*.

Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., & Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. *2012 IEEE International Conference on Robotics and Automation* (pp. 957–964).

Wu, H., Chen, S., Yang, B., & Chen, K. (2016). Feedback robust cubature Kalman filter for target tracking using an angle sensor. *Sensors*, *16*(5), 629.

Yoder, L., & Scherer, S. (2016). *Autonomous exploration for infrastructure modeling with a micro aerial vehicle* (pp. 427–440). Cham: Springer International Publishing.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section.

How to cite this article: Lutz P, Müller MG, Maier M, et al. ARDEA—An MAV with skills for future planetary missions. *J Field Robotics*. 2020;37:515–551. https://doi.org/10.1002/rob.21949