A Low-Cost Mobile Infrastructure for Compact Aerial Robots Under Supervision

Marc Lieser Henning Tjaden Robert Brylka Lasse Löffler Ulrich Schwanecke

***Abstract*— The availability of affordable Micro Aerial Vehicles (MAVs) opens up a whole new field of civil applications. We present an *Infrastructure for Compact Aerial Robots Under Supervision* (*ICARUS*) that realizes a scalable low-cost testbed for research in the area of MAVs starting at about $100. It combines hardware and software for tracking and computer-based control of multiple quadrotors. In combination with the usage of lightweight miniature off-the-shelf quadrotors our system provides a testbed that virtually can be used anywhere without the need of elaborate safety measures. We give an overview of the entire system, provide some implementation details as well as an evaluation and depict different applications based on our infrastructure such as an Unmanned Ground Vehicle (UGV) which in cooperation with a MAV can be utilized in Search and Rescue (SAR) operations and a multi-user interaction scenario with several MAVs.**

I. Introduction

Over the last years the quadrotor—a multirotor helicopter with four rotors—has emerged as the standard research platform for Micro Aerial Vehicles (MAVs) due to its affordability and mechanical simplicity. MAVs are utilized in numerous fields, e.g., first response, reconnaissance and surveillance, transportation and logistics, inspection of industrial facilities, measuring, construction and aerial photography or videography. In recent years, a lot of impressive research results have been published, such as quadrotors performing quite sophisticated feats with balls [1], [2], balancing inverted pendulums [3] and passing them to other quadrotors [4]. Further examples include quadrotors carrying out aggressive acrobatic maneuvers [5] as well as assembling structures with small construction elements [6].

The main prerequisite for autonomously controlling a quadrotor is the determination of its position and orientation in space, its pose. Outdoors, it can be determined based on information obtained from Global Positioning System (GPS) and on-board magnetometer and inertial measurement units. Indoors, i.e., in GPS-denied environments such as collapsed buildings, the pose has to be estimated using other sensors and algorithms. The most common one is an on-board camera in conjunction with Simultaneous Localization and Mapping (SLAM) algorithms as described in [7].

Research in the field of MAVs requires controlled indoor test environments, which at least consist of an (optical) outside-in pose estimation component as well as a radio control system to pilot the individual MAVs. Latest environments like the Flying Machine Arena (FMA) [8] or the General Robotics, Automation, Sensing & Perception (GRASP) testbed [9] are well-equipped with costly technology. Most test environments use multi-camera motion capturing systems, such as Vicon or OptiTrack for pose estimation, which settle in the six-figure US dollar range. A common used quadrotor is the Ascending Technologies Hummingbird, an MAV in the four-figure US dollar range of about $500\,\mathrm{g}$ weight.

The great expense of the aforementioned testbeds prevent them to become more widespread. But nowadays convenient hobby quadrotor platforms start at $10. Due to the size and weight of the latest consumer miniature quadrotors, elaborate security precautions which exceed the installation of fly screen are not necessary since these lightweight machines will not cause considerable damage to humans or components of the infrastructure. The hobby platforms are by far not as well equipped as professional quadrotors or even freely programmable. Therefore control tasks can not be accomplished directly on the MAVs but have to be done on the computer that remote controls them.

With *ICARUS*, we present an infrastructure for general research and application purposes consisting of a lightweight framework and software architecture resorting to affordable off-the-shelf hardware that is easily replicable. In its smallest expansion, our system is very portable and can be used for live demonstrations, e.g., as part of a lecture or exhibition. With *ICARUS*, research in the exciting field of multirotors gets started at overall hardware expenses of about $100 spent on a commercial multirotor platform and a radio control transmitter module, a Raspberry Pi in conjunction with its official camera and a tracking marker.

The following section gives an overview of our system and its components. Section III describes the software architecture. In Section IV the system's accuracy is discussed. Section V presents applications realized with *ICARUS*. Finally, Section VI summarizes our research and proposes future work.

II. System Components

An architectural overview of *ICARUS* is depicted in Fig. 1. As the main purpose of the presented system is to facilitate the realization of a low-cost testbed, we consequently use low-cost miniature off-the-shelf hobby quadrotors, which were slightly modified by attaching tracking markers. The quadrotors are tracked by an infrared-based monocular tracking system using at least one monochromatic camera. Thereby, the flight area can easily be expanded using multiple cameras. The quadrotors are piloted by a consumer notebook

The authors (`givenname.surname@cvmr.info`) are with the Computer Vision & Mixed Reality Group of RheinMain University of Applied Sciences, Wiesbaden, Germany, which supported this work by internal research funds.

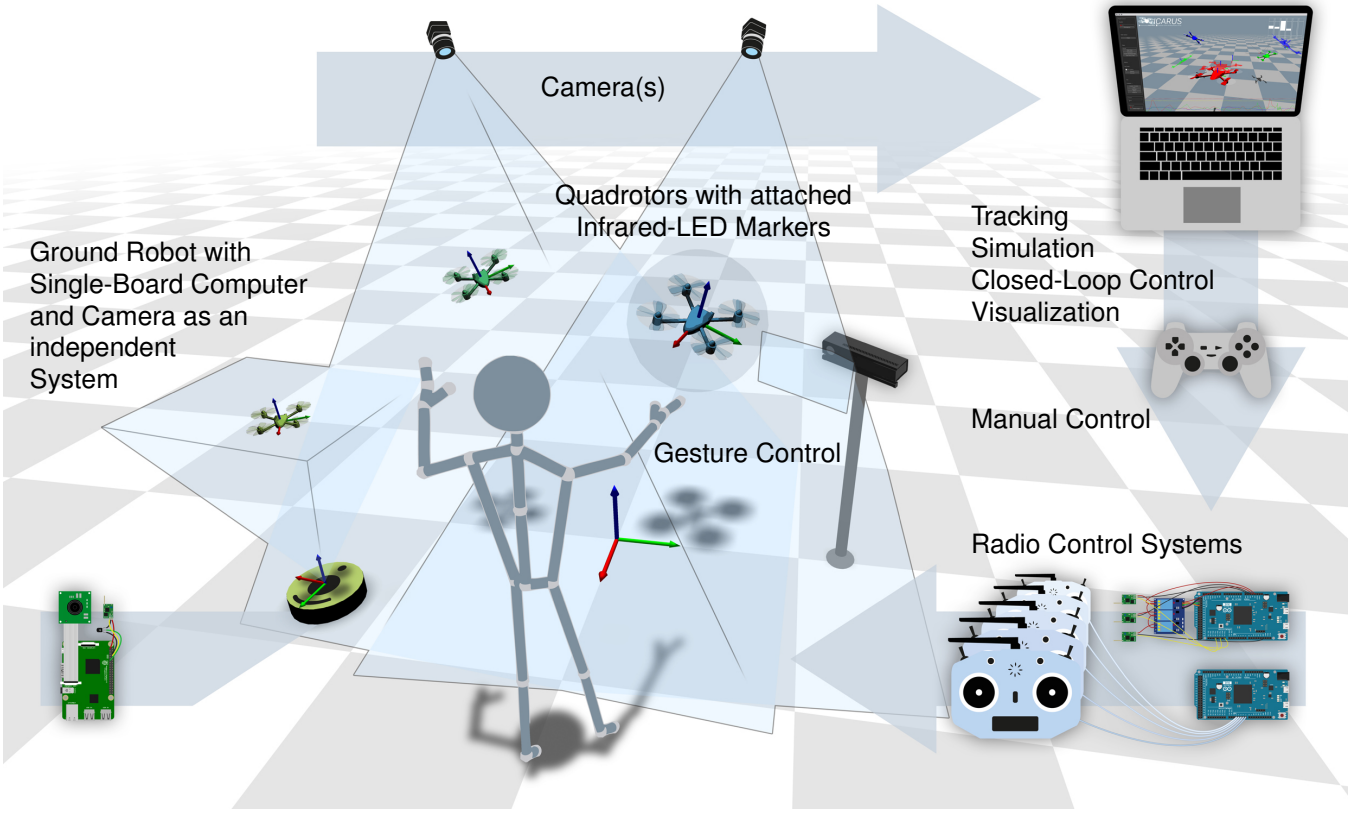


Fig. 1: Overview of *ICARUS*: After pose estimation of the individual quadrotors based on optical tracking the control variables are determined and sent via different radio control systems to the quadrotors.

connected to different radio control approaches. Override and manual control is enabled using a gamepad. The minimum setup can be realized for under $100 and therefore is affordable even for individual researchers and students. The complete system also runs independently on a ground robot as described later in Section V-A.

A. Optical Tracking

The main demands on the tracking of quadrotors are accuracy, frequency, robustness to rapid movement and the capability to distinguish multiple targets. In our system we use the High-Speed and Robust Monocular (HSRM)-Tracking system proposed in [10], a monocular outside-in tracking algorithm based on active infrared LED markers, which sufficiently meets all these demands while being very lightweight compared to multi-camera motion capturing systems used in other testbeds. One of the main advantages of this algorithm is, that it does not exploit any frame-to-frame strategies, guaranteeing prompt relocalization after temporary tracking losses, which is crucial in a MAV environment. Our prototype markers weigh about $4\,\mathrm{g}$, keeping the interference of the flight characteristics to a minimum.

Employing a monocular tracking algorithm, we can use one or multiple cameras, allowing us to easily scale the radius of action. The registration of multiple cameras to a common coordinate frame can be done by simply using one of the markers. Since each camera can estimate its relative pose to the marker, the relation between multiple cameras is instantly known whenever they see the same marker at the same time. The registration process determines the relative transformation between two cameras by minimizing the least mean squared error as proposed in [11] over a sequence of simultaneously estimated marker poses. Thus arbitrary multi-camera setups can quickly be calibrated incrementally when each camera's field of view overlaps with the field of view of at least one other camera. We fuse the measurements of all cameras over a parameterizable time window by linearly blending the marker poses in dual quaternion representation as described in [12]. Due to the monocular fashion of the tracking algorithm, multiple cameras with different frame rates can be utilized within the same setup.

Since the implementation of HSRM-Tracking is single-threaded, multi-camera setups can also easily be parallelized by simply processing each camera in a dedicated thread. Current quadcore consumer laptops could therefore process four high-speed cameras in parallel at full frame rate. Our most sophisticated implementation uses four high-speed Ximea MQ013MG-ON USB3 cameras in conjunction with Fujinon 1:1.2/6 mm DF6HA-1B lenses, operating at $150\,\mathrm{Hz}$ with 1280×1024 pixels resolution filming downwards from above the flight area. To be less dependent on ambient lighting, all cameras are equipped with infrared pass filters.

B. Quadrotor Control

We demonstrate the capability of our system using low-cost off-the-shelf miniature quadrotors from the hobby area. The only modification consists in the attachment of the aforementioned active tracking markers and connecting them to the quadrotors' power supply. All quadrotors possess on-board microcontrollers realizing an inner control loop to stabilize their horizontal position. Our system currently uses two approaches for the outer control loop: A simple hover controller which is tuned to maximize stiffness and a trajectory controller, both using a typical feedback control realized by multiple PID controllers. In section IV, we demonstrate that this simple control strategy leads to satisfactory results despite our tracking system refraining from using any prediction or estimation algorithms. Of course, these results can be further improved by the implementation of statistical filtering, such as an extended Kalman filter.

To apply the general PID control approach to our infrastructure, the terms of the PID controller have to be chosen and their gains have to be tuned. PID control is a compromise of different purposes and dependent on the demands to the system's dynamics. Generally simulations are helpful to test PID gains before applying them to the real system, thus reactions of the system to influences like changing the set point can be judged. All PID gains were determined experimentally.

In **hover control**, four independent PID controllers are used to approach the target pose. Thereby, each of the quadrotors input channels namely throttle, roll, pitch, and yaw are controlled the same way a human would do using a radio control system.

For the *throttle* control all terms of the PID controller are used. The height error is fed into the throttle controller. Next to the usual steady state error, the integral term also deals with the throttle needed to overcome gravity and compensates for the battery's dropping voltage. Relying on the integral term instead of a feed-forward term may result in inertia but yielded good experimental results. To avoid integrator windup, our system uses the conditional integration approach described in [13].

Due to our system's limitation of the quadrotors operating at a near hover state, the control errors for *roll* and *pitch* are calculated by projecting the quadrotors' positions onto the ground plane. For the quadrotor control to be independent of its yaw orientation the set point is transformed into the quadrotors coordinate system. Although a proportional and differential action would be sufficient for both the controllers, a low integral action was added to reduce the drift of hobby quadrotors, which would be leveled out by a hobby pilot by trimming the radio control.

Since our implementation is based on quaternions, the *yaw* control error can simply be calculated by multiplying the target orientation by the inverse of the quadrotor's heading and extracting the yaw Euler angle as the error fed into the control equation. As for the yaw controller a pure proportional controller proved to be effective.

The **trajectory controller** currently uses the approach proposed in [14], where next to the throttle and yaw PID controller described above two further PID controllers minimize errors along the path segment of the trajectory and orthogonal to it.

C. Radio Control Systems

In a regular handheld radio control, potentiometers pick up the positions of the levers and switches and convert them into electrical signals which are modulated to a high-frequency signal and transmitted to the on-board receiver of the quadrotor. After demodulation, the signal is converted into a proportional electrical signal controlling actuators such as electronic speed controllers or servos.

We substitute the handheld radio control system by a microcontroller connected to a computer. This microcontroller generates the required electrical signals to drive a transmitter based on control signals received from the computer. Thereby, the transmitter can be either a bare transmitter module (Arduino Triple Transmission (ATT) approach) or a handheld radio control connected to the microcontroller by its trainer port (Arduino Trainer Port (ATP) approach). Both approaches are open sourced at GitHub[1] and provide cheap solutions to computer-control multirotors or planes from the hobby area. The currently used hardware uses Spektrum DSMX modulation–which is widespread in the model flight community–and can be exchanged for devices using other hobby standards such as Graupner HoTT or FrSky ACCST. In addition to these approaches, *ICARUS* also supports the Bitcraze Crazyradio. All systems are enabled to be flown simultaneously. In the following we describe the implementation of both approaches based on the open source Arduino platform.

1) Arduino Triple Transmission: The ATT approach directly drives multiple transmitter modules removed from Spektrum budget radio controls via serial communication. As microcontroller we use an Arduino Due, which operates on $3.3\,\mathrm{V}$ matching the operating voltage of the transmitting units. Beside the USB interface it has three additional hardware serial pins allowing the control of up to three transmitter modules. To bind a module to a quadrotor's receiver the binding signal has to be applied before switching on the transmitter module. The required current of $30\,\mathrm{mA}$ cannot be supplied by the Arduino's pins, so some additional circuit based on a transistor or relay is required to switch the units on and off.

2) Arduino Trainer Port: The ATP approach employs the pulse-position modulated (PPM) trainer port signal and thus can be used with any radio control system equipped with a trainer port. In combination with an open source radio control, where manufacturer-dependent transmitters can simply be exchanged, there is no limitation to a particular radio protocol. As our system should be cost-efficient, we aim to drive as many control units as possible with a single microcontroller.

[1]Repository links listed at `cvmr.info/icarus/#opensource`.

As can be seen in Fig. 2, the frame of a typical trainer port signal takes 22 ms and encodes six channels: throttle, roll, pitch, yaw, and two additional channels (e.g., for switching flight modes). The length of a channel's high level signal depends on the according lever's position and takes 700 µs to 1530 µs. With a separation of each channel by a 400 µs low level signal, encoding all channels requires a period of 7000 µs to 11 980 µs. The rest of the frame is filled with a high level signal.

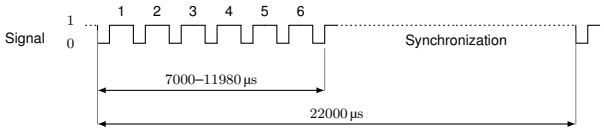


Fig. 2: Characteristic of a six channel trainer port signal.

A total number of 12 Bytes (2 Bytes per channel) are needed to encode control information within the signal. Limited by the maximum buffer size of 64 Bytes of the serial interface, up to five control units can be driven simultaneously by a single Arduino Due. A pulse-position modulated signal is typically created by using a sleep system call. Since these calls are implemented in a synchronized fashion, they cannot be used to create multiple signals in parallel on a single core architecture like Arduino. Our solution discretizes all five signals with a precision of one microsecond, in which within the synchronization phase a byte pattern mirroring the current signal states is created and then transferred to the Arduino's port registers at once. Due to the time needed for processing, we can drive all control units with a temporal accuracy of ±2 ms, which is sufficient for all control purposes.

III. Software Architecture

Our infrastructure is based on the libraries developed in [15] and was distributed and extended by the integration of an existing simulation environment in [16]. To achieve a maximum flexibility, the main demands of the *ICARUS* software architecture are scalability and platform independence in a distributed system. The developed software architecture connecting the parts of the infrastructure mentioned in the last section is displayed in Fig. 3 and uses a publish-subscribe messaging pattern to gain the desired flexibility. By providing loose coupling and a dynamic network topology the publish-subscribe pattern meets all requirements to the infrastructure. The following section describes the individual components of the software architecture.

The `ICARUS-COP` (Control Panel) is a web application which summarizes all important controls the user needs for interaction with the infrastructure. It enables the user to toggle radios and bind quadrotors and to execute predefined series of `Tasks` or to script new ones. The quadrotor's PID gains can be tuned with this interface as well. The origin of the reference coordinate system can be redefined by the current pose of a quadrotor. Beyond that, the user can manually control multirotors using a gamepad. The input generates `Commands` which then are sent to the corresponding radio control system. When using the `ICARUS-COP` on a mobile device its gyroscopes and accelerometers—if available—can be utilized for a more abstract control. The `Abstract Control` interface allows the user to pilot quadrotors by natural interaction. Currently this interface is implemented as gesture-based control and generates `Tasks` based on detected gestures. This interface is easily exchangeable for other interaction paradigms, e.g., a speech recognition component. The `Visualization` is a local OpenGL application providing visual feedback for the user by displaying the quadrotors' poses, their target poses, the control errors fed to the PID control as well as the camera poses, their frustums and images.

The `Flight Control` maintains synchronized task queues for each quadrotor. The PID control is encapsulated in a variety of yet implemented tasks, e.g., the hover task, which uses the tracking-determined pose of a quadrotor and calculates the control commands based on the PID control loop. The `Commands` then are sent to the radio bound to the according quadrotor. Beyond that, the `Flight Control` supervises each quadrotor and initiates emergency landing tasks in case a quadrotor leaves the radius of action.

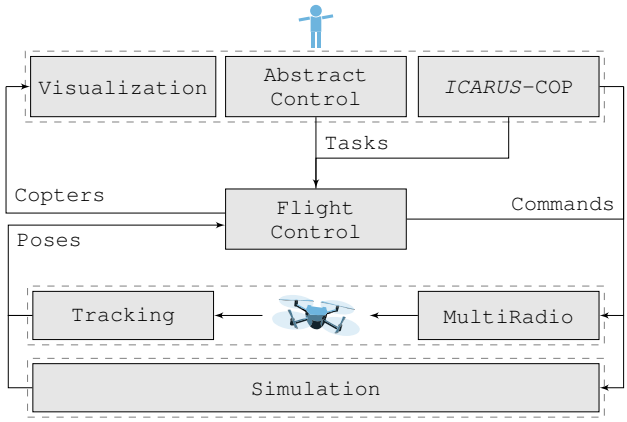


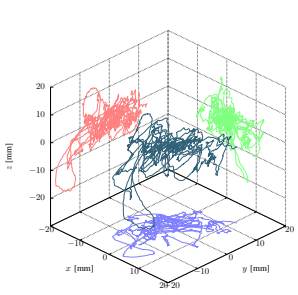
Fig. 3: The software architecture of *ICARUS*.

Due to the modularity in software architecture, components of the system can easily be replaced, e.g., pose estimation and radio control can be exchanged for a `Simulation` environment currently implemented by the Gazebo robotics simulator. The PID gains can be determined using the simulation, before applying them to the real quadrotors to prevent crashes. Using the same interfaces, simulated and real quadrotors can fly together in a mixed environment. Since the space of an outside-in controlled infrastructure is limited, the simulation environment additionally enables larger experiments, e.g., the generation of swarm scenarios which exceed the number of available quadrotors.

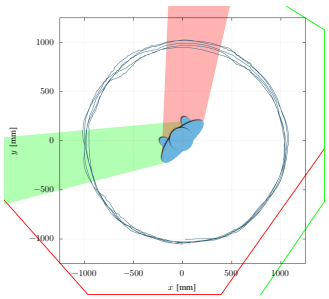
IV. Evaluation

This section evaluates the accuracy of our infrastructure in different flight modes. Videos of the experiments can be

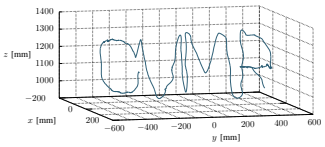
found online[2]. Evaluation periods of these experiments start with the arrival at the first waypoint and end with the arrival at the last waypoint. The poses provided by our tracking system were smoothed over a time window of 66.6 ms, which at a frame rate of 150 Hz corresponds to 10 poses in the single-camera-setup and 20 poses in the two-camera-setup. The precision of HSRM-Tracking (see [10] for details) has to be considered in the evaluation.



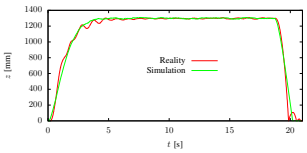
(a) 3D plot and 2D projections of the quadrotor's error in position during a 60 s hover flight.



(b) Flight of five circles with an obstacle at the center. The red and green lines depict the camera frustums with the red and green areas not visible for the according camera.



(c) Flight path following a prescribed trajectory.



(d) Comparison of a real quadrotor and its simplified model simulated by Gazebo.

Fig. 4: Results of different experiments.

A. Hovering

A flight in which the quadrotor was instructed to hover at the target height of 1300 mm for 60 s was carried out in a single camera environment. The target position of the quadrotor corresponded to a distance of approximately 1200 mm to the camera. The position error of the hover controller can be seen in Fig. 4a. The Root-Mean-Square Error (RMSE) of this experiment is 7.6 mm, the mean and standard deviations are $\bar{\mathbf{t}} = (0.6 \pm 4.4, -0.2 \pm 5.9, -2.3 \pm 4.1)$, and the maximum deviations are $\mathbf{d}_{\max} = (12.6, 18.3, 20.8)$.

B. Circular Trajectory

Using the trajectory controller, in this experiment a quadrotor was instructed to fly five circles with a diameter of 2000 mm at a speed of $0.5\,\mathrm{m\,s^{-1}}$. The distance between the waypoints was 25 mm. To be able to monitor this area, a two-camera-setup was used. To demonstrate the monocular fashion of our tracking algorithm, the quadrotor orbited a person occluding different areas of each camera's field of view, as can be seen in the long-exposure photography in Fig. 5c. In this experiment, the quadrotor flew with a mean velocity of $0.47\,\mathrm{m\,s^{-1}}$. The RMSE is 33.8 mm. The plot can be seen in Fig. 4b.

[2]Videos available at `cvmr.info/icarus/#videos`.

C. Skywriting

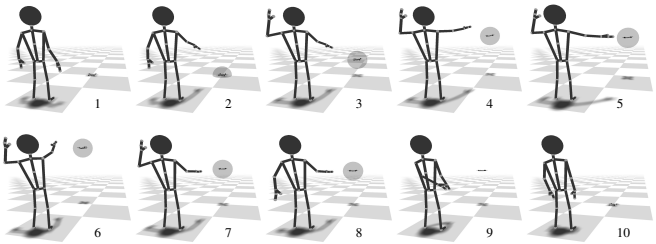
In this experiment, a quadrotor followed the trajectory of the letters CVMR with a total width of 1000 mm and a height of 292 mm using the hover controller. The baseline of the lettering was at a height of 1000 mm. A 3D plot of the quadrotor's trajectory is shown in Fig. 4c. The RMSE of this experiment is 22.2 mm.

D. Simulation

To compare a simulated and a real quadrotor against each other, a 15 s hover flight at the target height of 1300 mm was carried out in a two camera setup with both quadrotors using the same PID gains. The comparison of the heights depicted in Fig. 4d shows high consistency between simulation and reality.

V. Applications

As proof of concept, this section describes three applications based on *ICARUS*. These contain a quadrotor controlled by a ground robot, gesture-based human interaction with quadrotors and light paintings created by the quadrotors' trajectories.



(a) Gesture-based interaction: A user selects a quadrotor by pointing at it with the right arm (2). Lifting the left arm instructs the quadrotor to take off (3). While the left arm is up, the quadrotor imitates the motion of the user's right hand (4–7). When lowering the left arm, the quadrotor stays in hover mode and may be selected by another user (8). Bringing both hands together (9) lets the quadrotor perform a landing (10).

(b) MAV tracked and controlled by a mobile ground robot.

(c) Light paintings of the flights evaluated in Section IV.

Fig. 5: Different applications based on the presented infrastructure.

A. Flying Periscope

Our experimental Search and Rescue (SAR) application is based on a mobile ground platform and molds a flying periscope by collaborating with a quadrotor similar to the system proposed in [17]. A quadrotor controlled by the ground robot can be seen in Fig. 5b. Especially in the field of SAR operations the limited field of view of on-board sensors and cameras mounted on an Unmanned Ground Vehicle (UGV) can easily be expanded by collaborating with MAVs, which can investigate the surrounding area from a higher point of view. Furthermore the extended mobility of an MAV can be exploited. On the other hand, the UGV can serve the MAV as power supply platform to recharge its battery. For this project, we employed the whole infrastructure including HSRM-Tracking on the Raspberry Pi 3 using the Raspberry Pi camera module, operating at $50\,\mathrm{Hz}$ with 1280×720 pixels resolution.

B. Multi-User Interaction with MAVs

Since operating control levers is not very intuitive for the user, we extended *ICARUS* by a multi-user environment for gesture-based interaction and control of multiple quadrotors using a Microsoft Kinect. A typical user interaction sequence is shown in Fig. 5a.

C. Light Painting

The long-exposure photographies of the evaluation flights described in Section IV can be seen in Fig. 5c. This photographic technique is called light painting and provides a good intuition for the accuracy of our control system.

VI. Conclusion and Future Work

In this paper, we presented *ICARUS*, a least cost and portable infrastructure solely built on off-the-shelf hardware for the navigation and control of multirotors, especially of low-cost miniature quadrotors. We evaluated the components of the system with regard to pose accuracy and ported the infrastructure to a common mobile ground platform.

Future work regarding that mobile platform includes the integration into our main testbed to provide our quadrotors a landing platform equipped with inductive charging units in order to increase the radius of operation. Furthermore it is planned to extend our gesture-based control by exploring new human-robot-interaction paradigms such as tactile control of quadrotors as proposed in [18]. To enhance our infrastructure, we are currently working on the generation of feasible smooth trajectories and a model-predictive control approach to reduce the jerk of the quadrotors as described in [19]. We plan to make our implementation of the trajectory generation, quadrotor dynamics and control publicly available to the community by integrating it into ROS [20]. After that, it is our goal to escape the controlled indoor environment we are currently restricted to due to the use of an outside-in tracking approach. Thus each MAV will be equipped with a camera allowing for visual SLAM (e.g., ORB-SLAM proposed in [21], available in ROS), to be employed in order to obtain full autonomy.

Acknowledgement

The authors would like to thank Daniel Andres Lopez for his contribution to the Flying Periscope and Annalena Gutheil for implementing gesture-based interaction.

References

[1] M. Muller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 5113–5120.

[2] R. Ritz, M. Mueller, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4972–4978.

[3] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Robotics and Automation (ICRA)*, 2011, pp. 763–770.

[4] D. Brescianini, M. Hehn, and R. D'Andrea, "Quadrocopter pole acrobatics," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 3472–3479.

[5] S. Lupashin, A. Schoellig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA)*, 2010, pp. 1642–1648.

[6] J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, and M. Kohler, "Aerial Robotic Construction: Towards a New Field of Architectural Research," *IJAC*, vol. 10, pp. 439–460, 2012.

[7] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646–1656, 2014.

[8] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, 2014.

[9] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *Robotics Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, Sept 2010.

[10] H. Tjaden, F. Stein, E. Schömer, and U. Schwanecke, "High-speed and robust monocular tracking," in *10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, May 2015.

[11] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, Apr 1991.

[12] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Transactions on Graphics*, vol. 27, no. 4, pp. 1–23, 2008.

[13] K. Åström and T. Hägglund, *Advanced PID Control*. ISA-The Instrumentation, Systems, and Automation Society, 2006.

[14] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, pp. 1–14.

[15] M. Lieser, "Outside-in Tracking-basierte Regelung von Multicoptern," Master's Thesis, RheinMain University of Applied Sciences, Wiesbaden, Germany, May 2014.

[16] L. Löffler, "Entwicklung einer Infrastruktur für die Steuerung und Simulation von Multicoptern," Master's Thesis, RheinMain University of Applied Sciences, Wiesbaden, Germany, December 2015.

[17] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A monocular pose estimation system based on infrared leds," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 907–913.

[18] A. Gomes, C. Rubens, S. Braley, and R. Vertegaal, "Bitdrones: Towards using 3d nanocopter displays as interactive self-levitating programmable matter," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. ACM, 2016, pp. 770–780.

[19] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, Dec 2015.

[20] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[21] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.