Mechatronics 24 (2014) 41–54

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

A platform for aerial robotics research and demonstration: The Flying Machine Arena

Sergei Lupashin *, Markus Hehn, Mark W. Mueller, Angela P. Schoellig, Michael Sherback, Raffaello D'Andrea

ETH Zurich, Institute for Dynamic Systems and Control, Sonneggstrasse 3, ML K 36.2, Zurich, ZH 8092, Switzerland

ARTICLE INFO

Article history:
Received 15 November 2012
Accepted 15 November 2013
Available online 10 January 2014

Keywords:
Aerial robotics
Quadrocopters
Robotics testbeds
Networked control systems

ABSTRACT

The Flying Machine Arena is a platform for experiments and demonstrations with fleets of small flying vehicles. It utilizes a distributed, modular architecture linked by robust communication layers. An estimation and control framework along with built-in system protection components enable prototyping of new control systems concepts and implementation of novel demonstrations. More recently, a mobile version has been featured at several eminent public events. We describe the architecture of the Arena from the viewpoint of system robustness and its capability as a dual-purpose research and demonstration platform.

1. Introduction

Dedicated multi-vehicle aerial robotics test beds such as [1–3] first appeared around a decade ago – the MIT Raven [1] test bed being one of the first described in literature. A number of research results have been enabled by such test beds such as aggressive maneuvers [4], aerobatics [5,6], object manipulation [7,8], coordinated construction [9], and others, expanding the field of aerial robotics at an impressive pace. Traditionally limited to scientific experiments, these systems are now reaching wider audiences at installations such as the New Directors' Showcase at Cannes [10], the outdoor aerial light show at Ars Electronica 2012 [11], and our own shows further detailed below.

Since 2007 we have been developing an aerial robotics test bed called the Flying Machine Arena (FMA) (Fig. 1). Our goal is to create software and hardware infrastructure reliable and robust enough for regular public demos while also being sufficiently flexible for research use. Given the research context, it is natural that a system like the FMA undergoes continuous modification; yet to enable public demonstration the FMA also has to meet the conflicting requirement of being a reliable, predictable system, operating on demand and with little forewarning. To meet these opposing requirements, the Arena has been created as a strictly modularized platform where flight-proven components are used for demonstrations as a complete flight-ready system, or, for research, in near-arbitrary combination with new, malleable experimental modules.

The FMA infrastructure is mostly vehicle-agnostic and a variety of dynamic systems have taken advantage of the platform. However, typical public FMA demos use small quadrocopters, mainly because of their unique combination of simplicity, robustness, and agility. Therefore algorithms and methodology in this work will be presented in a generalized manner, though quadrocopters will be used regularly to describe concrete details, with necessary specific equations listed in the appendices.

In addition to the quadrocopter-specific work, the FMA has found use in various other projects [12–14], mainly as a (i) distributed rapid-prototyping environment and for (ii) performance validation and evaluation. For example, a vision-based autonomous multicopter [14] was flown in the FMA to collect ground truth localization data, taking advantage of the protected space, the motion capture system, and the networking middleware to simultaneously collect data and to provide a backup failsafe controller in case the on-board autonomy failed.

In this work we take a different perspective from previous aerial robotic test bed publications [1–3] and describe the FMA system architecture (Section 2) and core components (Section 3) from the viewpoint of system robustness and dual-use capability. Note that here the term "robustness" refers to the concept of resilience of a system as a whole under stress and non-ideal subsystem performance. Again with a focus on resilience, we include a brief high-level analysis of key system performance characteristics in Section 4.

Public demonstrations enforce system robustness while disseminating research results without requiring further dedicated and possibly distracting commitments such as challenges and

* Corresponding author. Tel.: +41 44 632 0608, mobile: +41 76 226 5145.
E-mail address: sergeil@ethz.ch (S. Lupashin).

0957-4158/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.mechatronics.2013.11.006

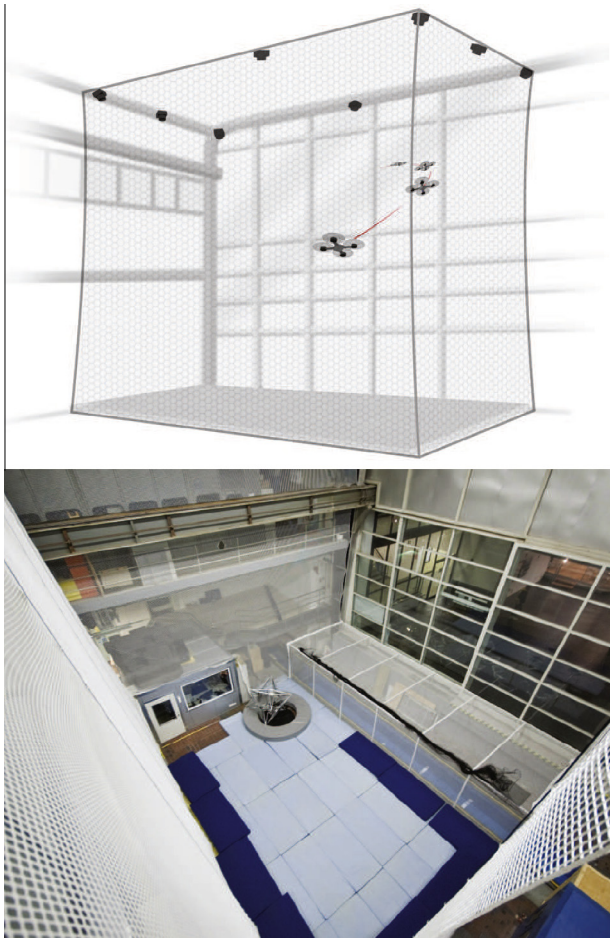

Fig. 1. The Flying Machine Arena at ETH Zurich.

competitions [15]. As both motivation and proof of its robustness, we believe the platform is unique in the number of regular public demonstrations conducted (e.g. over 100 events in 2011–2012). To this end, we describe the operation of the system, with focus on "graceful degradation"-type behavior under various non-idealities and subsystem outages (Section 5). To provide a better sense of how the FMA components are used in real life, we briefly describe the regularly performed FMA demonstrations in Section 6.

Another aspect described in this work is the FMA as a mobile platform: in 2011 the FMA has been extended from a permanent dedicated 10 m × 10 m × 10 m installation in Zurich to a mobile installation that has been exhibited in Europe and North America. Setting up the system in multiple locations and under pressure of public openings has taught us what processes and tools are important to building up such systems in a quick yet repeatable manner. We describe relevant details and experiences from our mobile exhibits in Section 7.

Finally, we conclude and summarize the presented work in Section 8 and present brief, compact, quadrocopter-specific details in the appendices.

2. System overview

At the top level, the FMA is organized similarly to the MIT Raven [1] and the UPenn GRASP test beds [3]. The combination of global sensing and off-board computing has been used widely in multi-vehicle research in the past, for example in Small-Size League RoboCup systems and derivatives [16,17] and in multi-vehicle test beds such as the UIUC HotDeC hovercraft test bed [18].

The data flow in the FMA is as follows (Fig. 2): vehicle/object pose measurements are provided by a motion capture system to software modules running on one or more standard computers running consumer operating systems. Within task-specific modules ("user code") and the *Copilot* safety/utility module, estimation and control pipelines produce vehicle motion commands. The *Copilot* picks which commands to issue based on failure detection code and the appropriate commands are transmitted to the vehicles. On board the vehicles, high-frequency controllers track these commands using on-board inertial sensors in feedback. All intermodule communication is via multicast UDP and the vehicles commands are sent over a dedicated wireless channel.

Note that the critical information flow between the components of the system is unidirectional. Bidirectional communication, e.g. telemetry from the vehicles, is supported, but is not required for controlled operation. All communication is done in a distributed, one-way manner, such that data sources are not affected by the corresponding listeners and there is no reliance on high-level code to keep track of the various components, preventing unnecessary interdependence. A set of core components with simple data dependencies (Fig. 2) makes up a robust position controller that can be depended on as a fall back during experiments and demonstrations.

2.1. Modularity and networking

The FMA is a distributed system, both at a conceptual and physical level: it consists of independently running computational processes that are arbitrarily distributed among one or more physical computing platforms linked together by either an Ethernet network or specialized industrial wireless channels (Fig. 3). Practically, the physical computing infrastructure ranges from a single Linux or Windows laptop to two to six or more networked laptop/desktop computers.

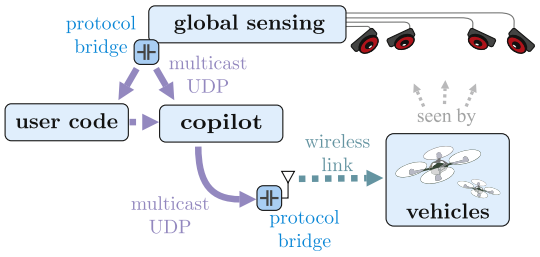

Fig. 2. The core control loop in the FMA. All data transmission is point to multipoint, available for any listeners – only the typical data flows are depicted.

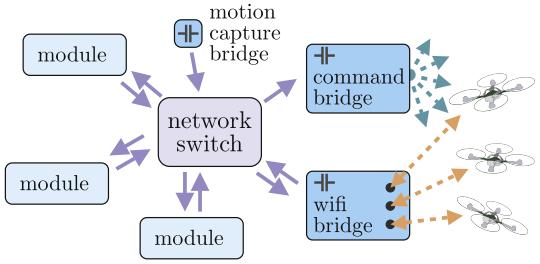

Fig. 3. Intermodule and wireless communication structure of the FMA. Solid lines represent multicast UDP data streams. Dashed lines represent wireless broadcasts (single-ended) and bidirectional connections (double-ended).

All communication is done in a "send once to anyone that listens, unreliably but often" manner. This has the advantage of low complexity in the transmitting and receiving software, and means that for lossy communication channels, data is never buffered or delayed due to retransmission attempts. There is no explicit clock synchronization in the system, but all sensor data is time-stamped against local hardware clocks, and whole-system communication latencies are carefully characterized (Section 4.1).

Intermodule communication is implemented in the form of multicast UDP streams, in a scheme evolved from the Cornell Urban Challenge framework [19]. Each packet type has a matching machine-readable blueprint, allowing for reception and processing both programmatically by software modules and through GUIs by users in real time. The multicast scheme is powerful in that there is no bookkeeping involved of who is using what information: drop rate tracking and basic filtering of duplicate/reordered packets are the only processing steps required on the client side, with no point to point communication or handshaking required for the system to function.

2.1.1. Logging and playback

As a convenient side effect, data logging and playback is trivial: for logging, a process subscribes to all of the multicast packets known to the system and records them, in order and time-stamped, to a file. To play back, these packets are read and resent, at the appropriate time intervals, back to the same multicast addresses, or tunneled to a specific host if the situation demands it. From the perspective of most software FMA modules, there is no difference between live operation, simulation, or playback: they simply subscribe to a multicast address and start receiving packets, if there are any.

2.1.2. Module inputs/outputs

In as much as practically possible, all intermodule communication associated with physical quantities is standardized as follows: module control inputs are physically meaningful, uncorrected quantities, and each module is responsible for making sure its behavior matches these ideal quantities, e.g. through calibration. Similarly, module outputs should be corrected, calibrated values. For example, rate gyro telemetry from the vehicles is compensated for bias and scaled to SI units, before being made available on the network.

Similarly, controller commands are given in equally physically meaningful terms: a zero attitude command means perfect alignment to gravity, even if the underlying control system actually hovers the vehicle at an angle to compensate for non-idealities. This approach creates a simple standard for intermodule communication involving physical quantities (ideal, SI units), and eases system-level debugging and monitoring.

2.2. Hardware abstraction and simulation

Software modules are isolated from hardware by protocol bridges that provide a system-wide hardware abstraction layer, allowing for substitution of hardware without any changes to algorithmic software modules. Only the corresponding bridges are changed. Furthermore, this allows for the substitution of a simulation environment in place of the physical system, enabling the testing of binary-identical software in simulated and real-world experiments (Fig. 4).

The simulation environment runs the on-board control, estimation and state machine code and uses a numerical integrator with the nominal equations of motion to simulate vehicle dynamics. Various parameters control the specifics of the simulation, such as disturbances, sensor noise, and simulation speed. The simulation environment is commonly extended to include application-specific physical objects such as chargers, balls and rackets.

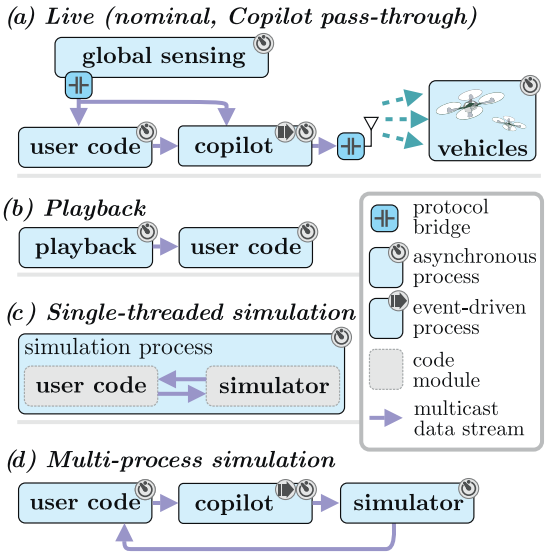

Fig. 4. Common FMA usage modes: (a) with the physical vehicles through the *Copilot* safety module, (b) playing back recorded data, (c) using a single-process repeatable simulation mode, and (d) running with an external simulator. The arrows represent the forward path of information through the system, though all processes can access any data; feedback loops and utility processes are omitted for clarity.

In practice, it is often desirable to carry out exactly repeatable simulations – something that is difficult to realize using an asynchronous multi-process simulation setup. For this reason the same simulation environment is also provided as a plug-in synchronous code module (Fig. 4c). In this setup, the application integrates the simulation modules directly in code, calling simulation functions on demand as part of a single processing loop. The simulation becomes synchronous, all random number generators in the simulation are configured to use a repeatable seed, and the simulated experiment is guaranteed to be exactly repeatable, allowing for precise, breakpointable analysis and debugging.

2.3. Vehicles

Demonstrations in the FMA typically use small quadrocopters based on the Ascending Technologies Hummingbird platform described in [20]. We use custom wireless communication and on-board electronics, assuring fine control over on-board estimation and control hardware and software.

A first-principles model of the quadrocopter dynamics is included for completeness in Appendix A and the on-board controller is described in Appendix B.

These vehicles have proven to be agile and reliable platforms; the typical operating quadrocopter in the FMA has sustained occasional hard crashes, in addition to scores of mattress-softened collisions, with the only observed consequences being propeller wear, occasional broken frames, rare motor failures, and several damaged gyros in the five-year operating history of the platform.

Other types (Fig. 5) of dynamic systems such as biplanes, tilt-rotors, the Distributed Flight Array [13], the Balancing Cube [12], and autonomous vision-guided multi-copters [14] have also used the FMA platform. All of these projects took advantage of the modular architecture of the FMA for data manipulation and logging, while some also used some of the vehicle-agnostic components such as the off-board estimation modules to quickly prototype control systems concepts.

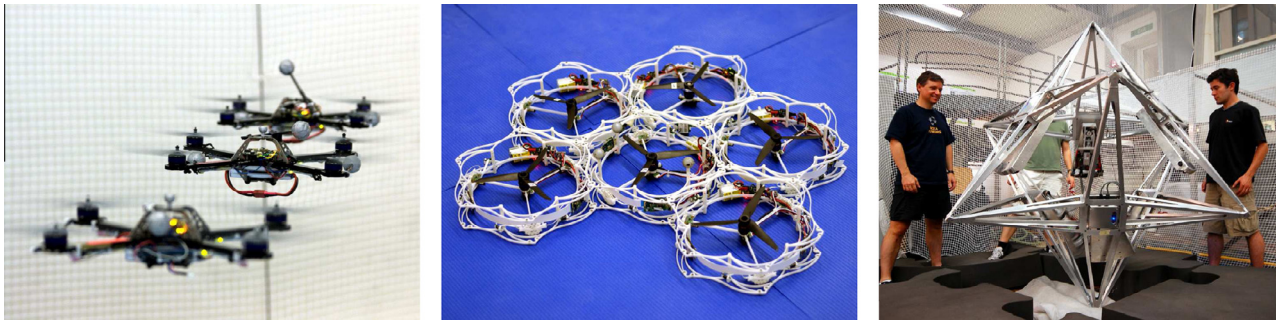

Fig. 5. While most work at the FMA uses quadrocopters (left), other dynamic systems such as the Distributed Flight Array [13] (center) or the Balancing Cube [12] (right) have also used the FMA infrastructure to accelerate development and/or collect ground truth data.

3. Core components

A small subset of the system must function reliably and robustly at all times to assure controlled operation. Failure of any one of these "core components" leads to emergency failsafe behavior, leading to a partial shutdown of the system and, if the situation is unrecoverable, an emergency stop of the vehicles (in case of flying vehicles, emergency descent, see Section 5.1).

The core components (Fig. 6) consist of global sensing (the motion capture system), the *Copilot* safety/utility module, the off-board estimation and control modules used by the *Copilot*, the wireless command interface, and on-board sensing, actuation, estimation and control.

3.1. Global sensing

We utilize a commercial motion capture system to measure the pose of marked objects in the FMA. For single-marker objects such as balls the information reported is the position of the object while attitude and position is reported for any rigid bodies made up of 3 or more markers. All measurements are relative to a user-defined "global" coordinate frame. The system is capable of producing measurements at a rates exceeding 300 Hz, though 200 Hz is typically used.

The permanent installation in Zurich uses eight 4-megapixel Vicon MX-F40 cameras. The angle of view of each camera is $66^\circ \times 52^\circ$. The mobile installation uses up to 20 Vicon T40 cameras, equipped with the same imaging sensor/optics. In both cases we use the standard near-infrared strobes in combination with retro-reflective 3–4 cm markers to track objects. This combination results in reliable marker detection at distances of up to 23 m away from the camera. In practice, we typically use a higher marker brightness rejection threshold (14 m detection distance) in order to reduce false detections. Although only two cameras need to see a marker to deduce its position, for robustness a minimum of three cameras is required.

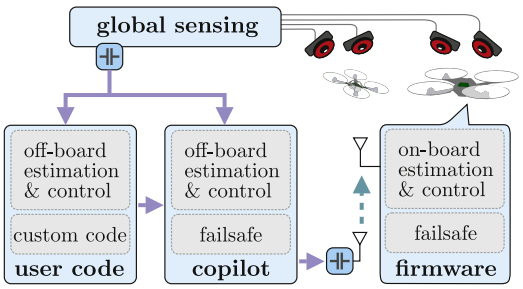

Fig. 6. Typical arrangement of core components as used for operation with physical vehicles. The arrows show the conceptual flow of information within the system.

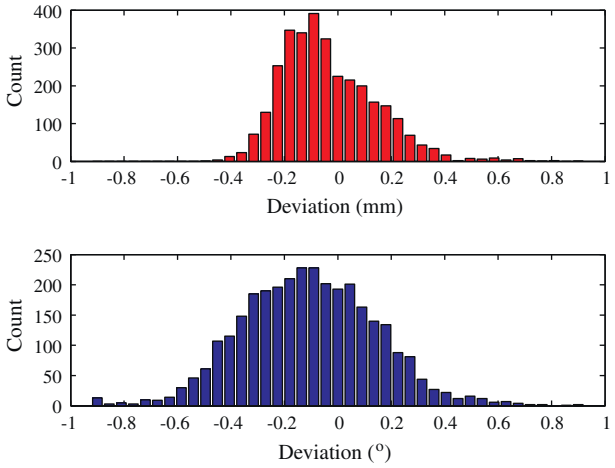

Fig. 7. Histogram representation of typical single-axis position and attitude measurement precision (data set of 3160 measurements) for a static 3-marker object in the 8-camera Zurich FMA system.

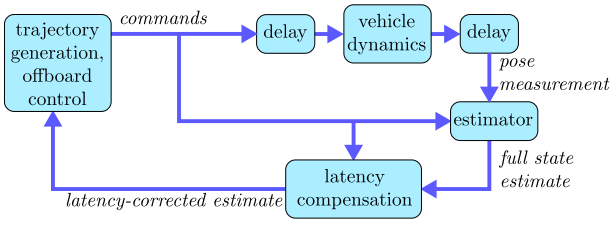

Fig. 8. High-level control and estimation components and relevant data flows in the FMA.

Fig. 7 shows the typical quality of localization data produced by the Vicon system as it is used in everyday experiments in Zurich.

3.2. Off-board estimation and control

Off board, the motion capture output and vehicle commands are fused into a latency-compensated full state estimate, to be used by high-level control algorithms. The flow of the off-board estimation and control is shown in Fig. 8.

3.2.1. Estimation

A predictor–corrector estimator uses a model of the system dynamics and commanded control inputs to generate an expected

state at the next measurement time, which is then fused with fresh measurement data to yield a full six degree-of-freedom (6DOF) state estimate.

To be more precise, let $\mathbf{x}[k]$ represent the 6DOF state of a vehicle at time t_k, consisting of the position, the velocity, an attitude represented by a quaternion and the angular velocity, with $\mathbf{u}[k]$ the corresponding command inputs. The nonlinear function $\mathbf{f}$ describes the nominal dynamics of the system, such that

$$\mathbf{x}[k+1] = \mathbf{f}(\mathbf{x}[k], \mathbf{u}[k]). \tag{1}$$

The motion capture system provides measurements of the vehicle's position and attitude. This is augmented with velocity and angular rate, derived by numeric differentiation, to yield $\tilde{\mathbf{x}}[k]$.

We denote our estimate of the state at time step k, using measurements up to time step j, by $\hat{\mathbf{x}}[k|j]$. The estimate update rule is then as follows:

$$\hat{\mathbf{x}}[k+1|k] = \mathbf{f}(\hat{\mathbf{x}}[k|k], \mathbf{u}[k]), \tag{2}$$

$$\hat{\mathbf{x}}[k+1|k+1] = \mathbf{C}\hat{\mathbf{x}}[k+1|k] + (\mathbf{I} - \mathbf{C})\tilde{\mathbf{x}}[k+1], \tag{3}$$

where $\mathbf{C}$ is a diagonal tuning factor matrix. The elements $\mathbf{C}_{ii} \in [0, 1]$ represent the relative weight of the prediction and measurements, and are tuned for by trial-and-error until acceptable performance is achieved.

While a Kalman filter-like implementation (such as the extended Kalman filter) might improve performance (depending on tuning), the above scheme is much simpler, avoiding partial derivatives, strong assumptions on process and measurement noise, and the issue of covariance adjustment in the face of quaternion renormalization. Furthermore, the presented scheme can be easily understood and extended for different vehicle dynamics by replacing the prediction function $\mathbf{f}$.

The use of an accurate nonlinear prediction implies that the estimation is robust to temporary measurement failures, such as when a marker is occluded. In such cases the estimator pushes forward the previous estimate using (2). Although this estimate will eventually diverge from the true state, the system will be able to continue operating in the face of short-term measurement blackouts.

By looking at the velocity and angular rate measurements (obtained through numerical differentiation), spurious measurements can be rejected. These outliers occur mostly when the motion capture system misidentifies an object in the space – due to e.g. specular reflections in the space, or poor calibration.

3.2.2. Latency compensation

We compensate for the known average system latency by predicting forward the vehicle state by this latency. The general strategy is to do all calculations on the state we expect when the current command arrives at the vehicle (see also [21]).

We denote the measurement latency as N discrete time steps and the control transmission latency as M steps. At time k the estimator described previously will operate on measurements up until $\tilde{\mathbf{x}}[k-N]$, and therefore yield an estimate $\hat{\mathbf{x}}[k-N|k-N]$. We store the $M+N$ most recent commands and apply them recursively to the vehicle's state using (2) to obtain $\hat{\mathbf{x}}[k+M|k-N]$, and use this to generate a control input $\mathbf{u}[k]$. Because of the M step communication delay, this control will actually arrive at the vehicle when its true state is $\mathbf{x}[k+M]$. This true state will equal the predicted estimate $\hat{\mathbf{x}}[k+M|k-N]$ in the case of perfect prediction.

Because we assume the system to be time invariant in (2), we do not need to know the values of M and N, but simply the total latency $M+N$, allowing us to design controllers without explicitly taking the latency into account. Assuming that there is no noise and our models are perfect, the above scheme will completely eliminate the effect of the latency. Fig. 9 shows this scheme in action for the attitude of a quadrocopter, where it can be seen that the latency-compensated angle correctly predicts the delayed measurement.

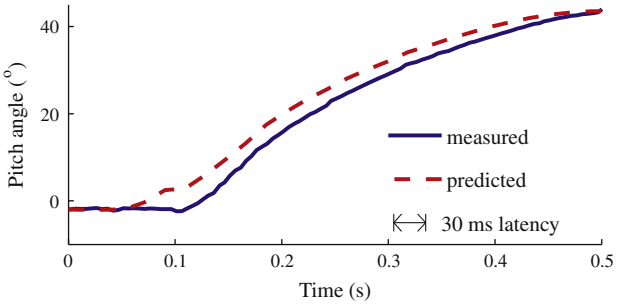

Fig. 9. Comparison of the measured vehicle pitch angle, and the estimator's latency-compensated prediction. Note that the prediction is done over the average latency, shown to scale in the legend.

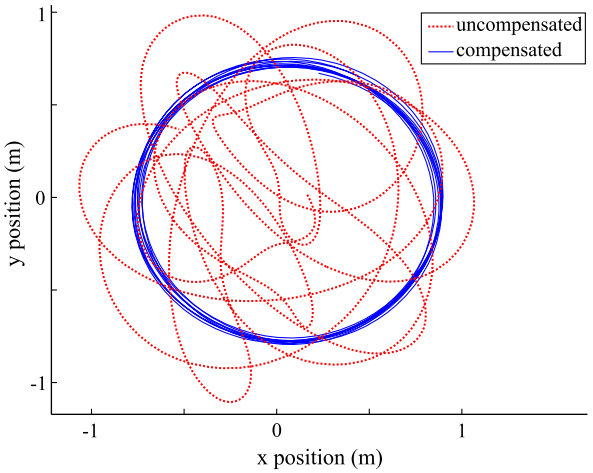

Fig. 10. Example of effect of latency compensation for aggressive maneuvers for a quadrocopter flying a circle.

The efficacy of this compensation is demonstrated in Fig. 10, where a quadrocopter was tasked with flying a horizontal circle of radius 1 m at a speed of 4 m/s. The quadrocopter was allowed 10 periods to settle into the trajectory, and then the mean squared error of the position deviation was measured over a period of 40 s. With the compensation this error was 0.273 m, compared to 0.806 m without the compensation.

3.2.3. Control

The control strategy is based on a cascaded loop shaping design strategy. The controller design is therefore split into the design of several controllers of lower-order dynamic systems. Each one of these lower-order systems is designed using a loop shaping feedback linearization approach, under the assumption that the underlying control loops react to set point changes without dynamics or delay. Individual control loops are shaped to respond to commands in the fashion of linear, time-invariant first- or second-order systems. The tuning parameters for the control loops consist of time constants and damping ratios, reducing the complexity of the tuning process and giving the user values that are intuitive to modify. Because only weak time scale separation exists between the cascaded loops, closed-loop performance and stability is verified by considering the entirety of the control system including all cascaded loops (see appendices). An example of this is presented in Section 4.2. This combination of simplified analysis used for tuning

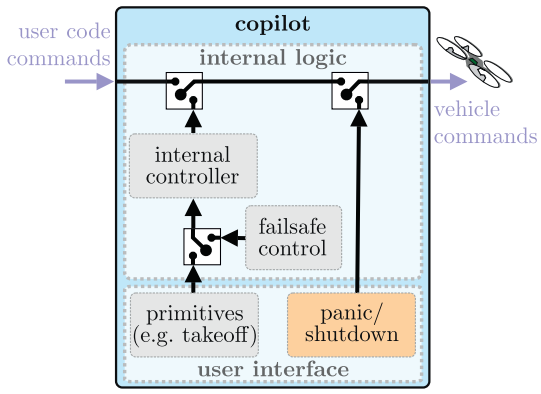

Fig. 11. The high-level decision logic of the *Copilot*.

and full model verification for stability permits us to design high-performance, yet tractable control laws.

3.3. Copilot safety and utility module

The *Copilot* program is usually used as a safety layer between an experimental application and the physical vehicles. It enables safety features such as vehicle control in case an external control process quits, crashes, or freezes unexpectedly, or in case of a user-initiated 'panic button' command (see Fig. 11). It also manages persistent system functions such as reference frame or vehicle calibration, and manages vehicle battery charge. To do this the *Copilot* uses the standard estimation and control components with a fixed known-stable set of parameters, in the case of flying vehicles to implement hovering, translation around the space to reach given positions, and takeoff/landing sequences.

A critical point is that any layers between an application and the vehicle may easily cause additional, variable latency. Because of this special care was taken in structuring the *Copilot* so that during nominal operation (that is, if all faults are clear, the vehicle is flying, and an application is providing vehicle commands), the *Copilot* forwards application commands, adding only an insignificant, fixed delay. If a fault occurs, if the application stops sending commands, or if a user commands the *Copilot* to take over control of the vehicle, it switches to a parallel-running internal estimator and controller and ignores further application commands until cleared to do so.

3.4. Wireless communication

Vehicles in the FMA almost always rely on wirelessly transmitted data for controlled operation; special care is therefore taken to fortify the system against potential wireless failures. We use two independent wireless systems to communicate with the vehicles: (i) a low-level, low-latency, unreliable, single-direction broadcast system for sending commands to the vehicles with minimum latency/overhead, and (2) a bidirectional, variable-latency, semi-reliable per-vehicle channel usually used for vehicle telemetry and parameter communication.

The dual wireless approach provides robustness-relevant benefits: firstly, the critical command link is as simple as possible and completely separate from more complex, bidirectional communication. Conveniently, this leads to a more predictable command latency behavior. Secondly, in case of wireless failure, we are able to choose between two physically independent channels to try to communicate with the vehicle.

The two wireless links may both operate in the 2.4 GHz band or in various other RF bands, depending on the configuration. The command wireless system uses frequency-hopping spread spectrum (FHSS) modules, while the bidirectional communications system uses direct-sequence spread spectrum 802.11b/g. FHSS radios "hop" many times per second within a part of the RF spectrum, meaning it is unlikely for the command signal to be blocked completely even in highly RF-saturated environments. This makes a FHSS radio ideal as a command channel, where moderate losses are tolerated in absence of a total communication blackout.

3.4.1. Command link

The command transmission system operates at the hopping frequency of the modules used (2.4 GHz Laird Technologies LT2510 or 915 MHz LT1110 or similar), e.g. 75.8 Hz for the LT2510. Each command is broadcast twice to decrease loss probability; the resulting typical loss rates are on the order of 1–5%. We observed that for typical flights there is little change to vehicle performance for losses up to 60% (Fig. 12).

Due to bandwidth constraints, a single command link currently supports up to 6 vehicles. However, since multiple command links can coexist in the same space and the FMA networking system allows for multiple coexisting bridges of the same type, there is a natural way to scale up this limit.

3.4.2. Telemetry/parameter link

A second, independent wireless system is used for non-critical communication such as feedback from the vehicle and on-board parameter read/writes. This wireless system uses 802.11b/g via the Roving Networks RN-131C modules, sending back packets at a rate of 80 Hz. Feedback includes battery voltage and current measurements, rate gyro measurements, the command loss rate, and other information useful for general diagnostics or for particular experiments. Parameter read/write communication is done using this wireless channel, via an acknowledge-retry scheme built on top of the UDP multicast communication framework.

3.5. On-board estimation and control

The on-board estimator and controller is tasked with using any locally available information (in this case, inertial measurements) in feedback to follow given commands by generating the appropriate commands for the motors. On vehicles used in the FMA the on-board estimation/control loop typically runs at 800 Hz. The main on-board processor is an NXP LPC2148 32-bit micro-controller and the on-board inertial sensing unit is an Invensense MPU-6050.

The angular velocities of the vehicle are estimated using a predictor–corrector algorithm. We assume that the vehicle's axes are decoupled. As an example, consider an estimate of the roll rate $\hat{p}[k]$ utilizing the gyro measurements $\tilde{p}[k]$ and the gyro bias estimate $\hat{p}^0$. The relative importance of the measurement is captured

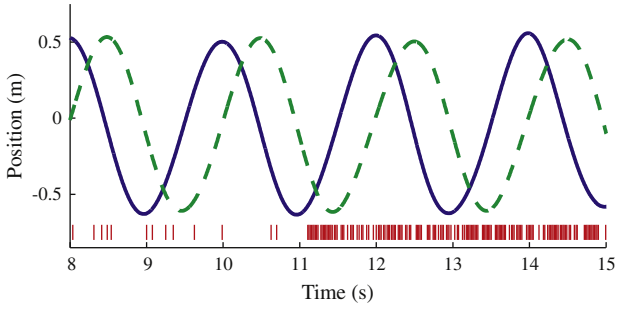

Fig. 12. Plot of x (solid) and y (dashed) vehicle coordinates of a quadrocopter flying a circle at 3 m/s during a command loss test. Each red line indicates a lost command. At 11 s the loss rate is artificially raised by 60% by probabilistically blocking packets. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

by $c_p \in [0,1]$. Prediction is done using the on-board sampling time Δt_{ob}, and the commanded moment around that axis – see (A.3).

$$\hat{p}[k+1|k] = \hat{p}[k|k] + \Delta t_{ob}\ l(\check{f}_2 - \check{f}_4)/\mathbf{J}_{xx},$$
$$\hat{p}[k+1|k+1] = c_p\hat{p}_1[k+1|k] + (1-c_p)(\tilde{p}[k] - \hat{p}^0). \quad (4)$$

The on-board control laws are designed in a similar fashion to the off-board control laws, using feedback linearization and loop shaping to nominally achieve dynamics of cascaded first-order systems with specified time constants. This approach allows the use of physically different vehicles (e.g. vehicles carrying extra load, or different in size) without modifications to the off-board control algorithms. The differences between vehicles are nominally compensated for by the on-board control algorithms.

It should be noted that, during nominal operation, the control loops running on board the vehicle only perform feedback control on states which are observable from local (i.e., inertial) measurements. This avoids the necessity to transmit accurately timed global sensing data to the vehicle data at high rates.

When the vehicle is in hover, on-board calibration performs sensor bias and motor performance estimation, allowing for higher overall vehicle performance (see Appendix C).

4. System analysis

In order to confirm the validity and effectiveness of the algorithmic approaches presented in the previous chapter, a number of experimental and analytic tests were carried out. In this chapter, we present parts of this analysis as a reference point on the flight performance achieved.

4.1. System latency

We have found latency in the system to be important to both repeatability and achievable performance. Communication latency from observing objects to vehicles receiving wireless commands is approximately 30 ms (Fig. 13), as measured by a light-detection test. To perform this test, an infrared LED, connected to a vehicle, is placed in the space. It is commanded to turn on through the command transmission system. When lit, the LED is seen by the motion capture system. The time between the command and the reception of a motion capture packet confirming that the light turned on is the measured full-loop communications latency of the system. Similarly, the on to off transition of the LED is also timed, as a check for intentional filtering/detection delays in the motion capture system.

Ignoring effects of wireless packet drops, the system communication latency varies between 22 and 46 ms. This is on the order of the expected delay: the 22 ms represents repeatable latency

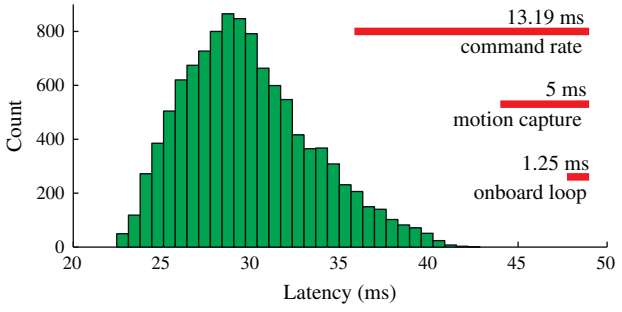

Fig. 13. Histogram of total system latency as measured by an infrared LED test (11,000 measurements). The time periods of the major independent processes in the FMA are shown for comparison.

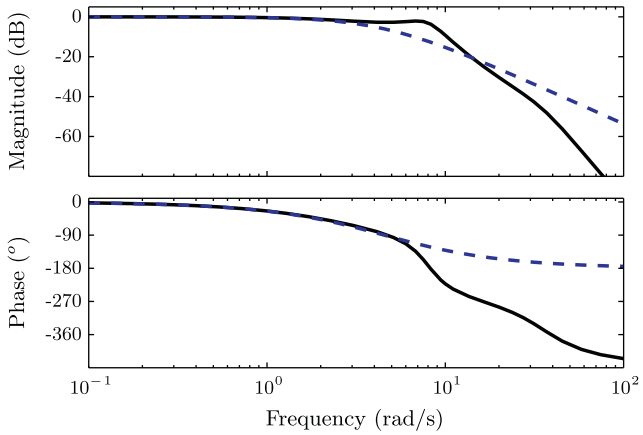

Fig. 14. Frequency response of all linearized cascaded feedback control loops (solid), compared to the loop shaping nominal design (dashed). It can be seen that the full (fifth-order) system follows the nominal design well for frequencies up to 5 rad/s, with the underlying loops influencing its behavior at higher frequencies.

encountered in every trial due to buffer transfers and accumulation delays, while the 24 ms varying component represents variable delays due to the independent asynchronous loops in the system: the known components consist of 13.19 ms for the period of the wireless transmission loop, 1.25 ms for the on-board control loop, and 5 ms for the motion capture system.

4.2. Cascaded control linear analysis

Controllers in the FMA rely on a strongly cascaded control scheme, as was introduced in Section 3.2.3.

To demonstrate the validity of the cascaded control loop design and its performance implications, we analytically evaluate the full cascaded closed-loop quadrocopter feedback control loop near hover for controller parameters as they are typically used in the FMA. We linearize all control loops about the hover point (defined by a constant position and yaw angle) and neglect discretization, latency, and estimator dynamics.

The frequency response of the resulting fifth-order system is shown in Fig. 14, along with the nominal second order system response that the loop shaping is designed for. It can be seen that, for frequencies up to about 5 rad/s, the system dynamics match the loop shaping design well. At high frequencies, interactions with the underlying control loops become significant, with the system response differing from the design.

5. Failure modes

We distinguish between graduated performance degradation (e.g. a proportion of the radio command packets are lost) and absolute outages (e.g. the radio channel as a whole is lost). In this section we will focus on failures of the second kind, i.e. absolute failures that render normal operation impossible. The modes were selected based on experiences with the system, and the mitigation strategy is designed to have a minimum impact on the operation of the system, and to be completely transparent to the user in normal operation.

5.1. Off-board outage failsafe

We consider two off-board outage failure modes: if the global sensing fails, the off-board system can no longer accurately estimate the state of the vehicles; while if the command wireless

channel fails, the off-board system can no longer command the vehicle. Because "do nothing and wait" is not a valid action for a flying vehicle (as opposed to e.g. a ground vehicle), an on-board emergency control was developed for the quadrocopters in the FMA, to mitigate the danger of an off-board outage [22]. The goal is to safely bring the quadrocopters to a standstill when there is no external data to rely on.

This is done by periodically sending to the quadrocopter the current off board state estimate over the bidirectional communication link. These state estimates are then pushed forward with measurements from the on-board rate gyroscopes (Eq. (4)), which are integrated to yield open-loop estimates for the vehicle attitude and velocity (Eq. (A.1)). Because we are integrating a noisy, biased signal, the angle and speed estimates will only be valid for short periods of time.

A prerequisite for entering emergency on-board control is that the on-board estimates for the quadrocopter's angles and lateral speeds are reasonable (not too large), and that these estimates have been recently updated with external data. If the estimates fail this test, the quadrocopter immediately switches off all propellers, the idea being that the worst case scenario when taking no action is better than that for executing an unpredictable action.

If an off-board outage occurs, and the quadrocopter's on-board state estimate is valid, the quadrocopter will attempt to achieve and maintain a hover for a fixed period (specifically, 3 s), using a linear near-hover controller. After this, the quadrocopter will perform a "crash landing", whereby it accelerates downwards at 1 m/s^2 for the time period required to reach the ground from its last known height. These stages are illustrated in Fig. 15, where part of a trajectory is shown for a quadrocopter starting with a horizontal speed of 1 m/s when an outage occurs.

If the vehicle is in emergency on-board control mode, but the original off-board outage is resolved (i.e. external communication is restored, or the off-board estimate is reliable again), we immediately switch back to off-board control. In these cases the *Copilot* enters a fault mode and commands the vehicle to hover at its current location, and waits for user action. More information on the emergency mode can be found in [22].

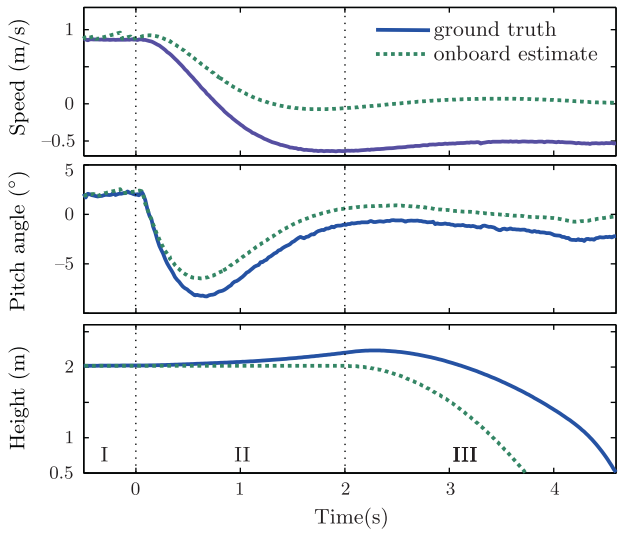

Fig. 15. An example of how the system responds to an off-board outage. Up to time zero (phase I) the vehicle is under external control. In phase II the vehicle attempts to maintain a hover using the emergency on-board controller, and executes a crash landing in phase III. The vehicle had a speed of 1 m/s when the outage occurred, and shown are the speed in the x-axis, and the pitch angle θ.

5.2. Low battery failsafe

Every healthy vehicle will eventually become uncontrollable once its charge or fuel runs out. To protect against this, and to increase system autonomy, we use a battery charge estimator that provides the approximate state of the charge for each vehicle while it is in flight. We use a simplified lithium polymer battery model with two internal RC elements. Based on this model and the on-board voltage and current measurement we use a Kalman filter similar to [23] to estimate the state of charge.

This estimator is run as part of the *Copilot*, with the resulting fuel level estimate available to internal fault-handling code (e.g. trigger landing on low threshold) and to other software modules.

6. Demonstrations

The research results enabled by the FMA have led to a number of regularly shown quadrocopter demonstrations (Fig. 16) that showcase the capabilities of the developed methods. All of these demonstrations build on the standard core components presented previously in this paper; in some cases these components are used as-is while in others they were extended to include new capabilities and features. We briefly present the demos commonly shown at the time of the writing of this article, to demonstrate the capabilities of the FMA and how it is used in actual research and demonstration projects.

6.1. Interaction with pointer/Kinect

Two demos enable visitors to quickly and intuitively control a vehicle flying in the FMA. The first approach uses a "pointer", consisting of a pointing device that is held by the user and tracked by the motion capture system. The direction of the pointer determines the desired position of the vehicle, and some simple gestures allow for commanding takeoff/landing and simple aerobatics (flips).

A more advanced interface, based on a student project [26], is provided by taking advantage of the Microsoft Kinect sensor: the sensor is labeled with markers and tracked by the motion capture system; this allows the Kinect's measurements to be transformed into the global FMA reference frame, allowing for direct interaction between a user that is tracked by the Kinect and a vehicle. A Kalman filter is used to filter user motion and to provide velocity (feed-forward) commands to the vehicle. The result is that the user's hand motion is closely imitated by the vehicle (Fig. 16d), times a scaling factor, enabling a three-dimensional, highly dynamic body control of the vehicle, in an experience that is somewhat akin to "force control" from the *Star Wars* franchise.

6.2. Adaptive aerobatics

This experiment demonstrates model-based policy-gradient learning for high-performance open-loop aerobatics [6]. In particular, a quadrocopter demonstrates automatic learning of single-, double-, or triple- flips (Fig. 16b) and aggressive translation maneuvers. This method requires a first-principles model and a way to observe the final state error of a maneuver, meaning that the exact trajectory of the physical aerobatic motion does not need to be known in advance.

6.3. Iterative motion learning

Another learning approach attempts to improve the tracking of a nominal trajectory throughout the motion. Using an Iterative Learning Control method [25], a given trajectory and model is transformed to a lifted domain representation. After a vehicle

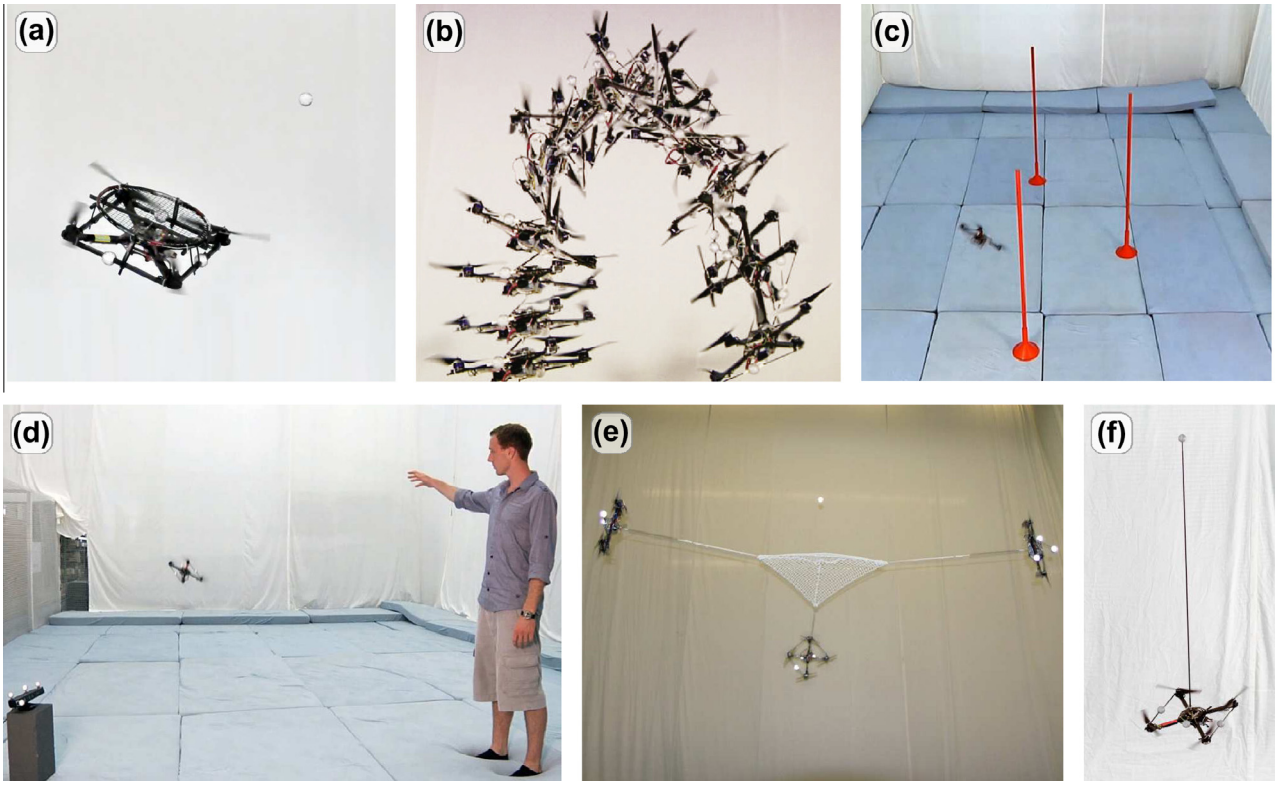

Fig. 16. Some of the quadrocopter demonstrations built on the FMA platform: (a) ball juggling [24], (b) adaptive acrobatics [6], (c) iterative motion learning [25], (d) interaction via Kinect [26], (e) cooperative ball throwing/catching [8] and (f) inverted pendulum [7].

attempts to fly the trajectory, its deviations are used to estimate the model error along the trajectory, which is used in an optimization to update trajectory feed-forward inputs. This allows the vehicle to improve its trajectory tracking from iteration to iteration. To demonstrate this method, a vehicle learns to fly through a near-arbitrary slalom course (Fig. 16c).

6.4. Ball juggling

One or two quadrocopters equipped with badminton rackets attempt to intercept and hit a ping-pong ball, covered in retro-reflective tape to be tracked by the motion capture system, to hit it towards each other or a target point (Fig. 16a) [24]. This involves a trajectory planning problem as the vehicles must be able to reach the intercept point at the right moment, with the right translational velocities and the proper attitude. All of these constraints have to be met with high precision and accuracy to enable a smooth "quadrocopter ping-pong" game. A learning layer built on top of the basic intercept code enables automatic learning of parameters such as ball drag and racket offsets.

6.5. Multi-vehicle transitions

This demonstration shows a sequential convex programming approach for generating collision-free trajectories for a group of quadrocopters attempting to translate through a shared airspace [27]. A number of quadrocopters (typically four to six) are flown in the space and given either a pre-programmed or random arrangement of waypoints. The multi-vehicle trajectory generation algorithm guides the vehicles to these waypoints, often across each other's paths, in a smooth (e.g. jerk-minimizing) manner. This demonstration also acts as a test case for transition code that is used in other projects, e.g. multi-vehicle dancing.

6.6. Cooperative ball throwing and catching

Three quadrocopters linked by a flexible triangular mesh attempt to throw and to capture a ball (Fig. 16e) [8]. The dynamics of the vehicles in this case are significantly different, since they are physically linked with stretchable material. The simulation environment was extended to allow the simulation of these maneuvers, and custom failsafe behavior was implemented, since the standard failsafe behavior leads to unpredictable results for the case of physically linked vehicles, though standard monitoring components, like the battery charge estimator, remained unchanged.

6.7. Construction of brick structures

A fleet of quadrocopters equipped with grippers construct a pre-specified structure from foam bricks in a coordinated fashion. More detail about this project is given in Section 7.1.

6.8. Multi-vehicle dancing

This project aims to demonstrate coordinated multi-vehicle flight performed to music [28]. One specific requirement in this application is high temporal accuracy in the trajectory tracking to synchronize motion to music. This project builds upon the estimation algorithms and the control schemes mentioned previously, allowing us to concentrate on higher-level challenges. For synchronization, an adaptation method is used to adjust the feed-forward input signal sent to the standard off-board trajectory tracking controller (Appendix D). By relying on the performance of the trajectory tracking controller, the multi-vehicle coordination problem at the core of this project is reduced to a trajectory planning problem, thereby simplifying the task at hand.

6.9. Flying inverted pendulum

This project aims to balance an inverted pendulum on a quadrocopter (Fig. 16f) [7]. Internally, it is implemented as a task-specific estimator and controller that interacts with the standard simulator and the communication infrastructure. In simulation, the vehicle dynamics are extended to include the equations of motion for the inverted pendulum. The balancing task has more recently been extended to allow a first vehicle to maneuver such that the pendulum is thrown in the air for a second vehicle to catch it and to balance it again [29].

6.10. Physical human–quadrocopter interaction

The application of admittance control to quadrocopters is demonstrated in this project [30]. The admittance controller allows a user to define the apparent inertia, damping, and stiffness of the robot, and interact with it by physically guiding it. The project uses a custom estimator to include the estimation of external forces acting on the vehicle, and leverages the standard trajectory tracking controller by adjusting the reference trajectory applied to it such that the desired admittance properties are achieved.

Fig. 17. A quadrocopter placing a foam brick as part of the FMA-enabled "Flight Assembled Architecture" performance at Orléans, France in 2011. The resulting structure is now part of the permanent collection of the FRAC Centre. (Photo: Francois Lauginie).

7. Mobile FMA and external exhibits

A mobile version of the FMA system was built for performances outside the lab. Physically, this is a set of laptops and a standard 19 in rack, housing two Vicon Giganet hubs for up to 20 Vicon cameras and a managed smart switch (for example, a Cisco SG300-series switch). The cameras are packaged in dedicated cases and the rack is shock-proofed, resulting in approximately twelve ship-ready cases, with a total weight of approximately 300 kg. When running with 19 cameras, the total system consumes approximately 600 W.

The current system has been used for flight spaces ranging in size from 185 m^3 to 720 m^3. The size of these spaces is largely defined by the available space at the venue in conjunction with the possible camera mounting points. In order to function properly, the motion capture cameras must remain rigidly fixed relative to each other, and, preferably, relative to the space. A good solution for this is a metal truss suspended from the ceiling, though other solutions, such as mounting the cameras directly to mount points on the ceiling/walls, have been used. Prior to arriving at the venue, a camera simulation tool is used to predict the space coverage of the motion capture system. The user then iterates over the camera positions and orientations until a satisfactory result is obtained, and defines the flight space based on the space coverage results.

The truss may also serve a dual purpose as a mounting point for safety nets, though in this case two trusses should be used to avoid people on the ground from moving the camera-truss assembly. Given a prepared suspended truss structure for the cameras and nets, it currently takes four people approximately 6 h to properly set up the mobile system, though this estimate depends on the specifics of the setup.

After the initial set up of the system, a calibration of the motion capture system is performed using a calibration wand (provided by the manufacturer of the motion capture system), mounted on a long, telescopic rod to allow the user to move the wand through the entire flight space. The calibration algorithm computes, amongst others, the position and attitude of each individual camera. This data is then fed back into the camera simulation tool to verify the predicted space coverage, and adjustments are made to the camera placement if necessary.

7.1. Flight-enabled architecture exhibit

The mobile FMA was used for a demonstration in Orléans, France in December 2011, where quadrocopters cooperatively constructed a foam brick tower (Fig. 17). The system to realize this

Fig. 18. The cooperative ball catching/throwing demo (Section 6.6) shown in front of a live audience at the 2012 Hannover Messe.

performance built upon the components presented in this work, with task-specific extensions such as specialized visualization modules, brick gripping and placement hardware and software modules, and a module for overall building coordination.

The system was deployed and tested in an unprepared space in a span of a week, at the conclusion of which we were able to demonstrate the system in front of the public, with showings spread over four days. We used four quadrocopters and four automated recharging stations to automatically maintain a controlled rate of build for the tower. 1500 bricks were placed with centimeter accuracy to build a 6 m-tall, 4 m-diameter tower.

Nineteen Vicon T-40 cameras were used to cover the space, carefully arranged to assure flight space coverage throughout the performance. Because the tower build ran through several days, and bricks had to be placed with high accuracy relative to the rest of the structure, special care was taken to maintain a consistent coordinate system, even as the cameras moved and drifted slightly over time. This was accomplished by tracking several fixed reference markers in the space and calculating appropriate transformations from the current motion capture coordinate system to the initial "ideal" one.

In addition, this demonstration was particularly challenging to our safety features, since the space was crowded with unprotected visitors. Due to various factors such as space lighting and reflectors/lights in the crowds, the motion capture software dropped a higher than usual number of data frames. In one case, the motion capture software crashed, causing a complete global positioning blackout; the failsafe features kicked in, automatically sounding a warning alarm. The quadrocopters maintained blind flight before descending giving the public time to react, turning a potentially disastrous situation into an inconvenient, but safe, glitch.

7.2. Other exhibits

The Flying Machine Arena was also shown at the 2012 Hannover Messe (Fig. 18), at the 2012 Google I/O at San Francisco, at the 2012 Zurich.Minds and at the 2013 Edinburgh TEDGlobal events. In these cases a 13-camera system was used to provide motion capture coverage. Each installation took 4–6 h to set up from shipped containers and 4 h to break down back to a packed form.

Leveraging the distributed architecture, additional laptops could be safely attached to the system at the last moment to run non-critical tasks. For example, we showed a live 3D visualization of the motion capture data to give visual feedback about the principles behind the system to the public. In a similar vein, we used the FMA plotting and real-time data manipulation tools to present live examples of concepts such as feedback control or sensor noise.

8. Conclusions and future work

In this work we have presented the Flying Machine Arena platform, including how robustness and reliability concerns affected the design of the system and how the system handles various failure modes. As test beds such as the FMA grow more complex, robustness and system-level design gain further importance. We hope that this work may be useful as a reference point to current and future designers, builders, and users of such platforms.

The demonstrations enabled by the FMA serve as both motivation and validation for the robustness of the system, even as many of the underlying components change day to day as part of ongoing research. We hope that in the long term FMA-enabled demonstrations and performances will help further the field of aerial robotics, especially in terms of reliability and system robustness.

We have omitted many of the more advanced features of the FMA from this work such as multi-vehicle air traffic control or a detailed description of the automatic high-rate charging stations. These tools act in different ways to increase the robustness of the system as well, but are more recent and less mature features that have only been used in isolated scenarios. We are also investigating ways of reducing our reliance upon the high frequency, high accuracy motion capture system by using different sensors and more sophisticated estimation techniques, in which case the existing FMA infrastructure is used to validate the proposed methods with ground truth data.

The safety/reliability features of the FMA are currently focused on failures occurring in communication or in the off-board software components. This is a reflection of our subjective experiences and the fact that the on-board code and hardware are relatively stable. We plan to investigate the treatment of hardware and on-board software failures in the future.

Acknowledgments

The authors thank the many collaborators on this project, especially Federico Augugliaro, Fabian Müller, and Thomas Kägi; a complete list of contributors can be found online at www.flyingmachinearena.org.

This research was supported in part by the Swiss National Science Foundation and through direct grants from ETH Zurich.

Appendix A. Quadrocopter dynamics

Translation in the global frame (Fig. A.19) is given by

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \mathbf{R}\begin{bmatrix}0\\0\\c\end{bmatrix} - \begin{bmatrix}0\\0\\g\end{bmatrix}, \tag{A.1}$$

where $\mathbf{R}$ is the rotation matrix from the body frame to the global reference frame, c is a mass-normalized acceleration due to thrust, and g is acceleration due to gravity.

The evolution of $\mathbf{R}$ is governed by the angular velocity $\boldsymbol{\omega} = (p, q, r)$ [31]:

$$\dot{\mathbf{R}} = \mathbf{R}\begin{bmatrix}0 & -r & q\\ r & 0 & -p\\ -q & p & 0\end{bmatrix}. \tag{A.2}$$

The body rates as well as c evolve according to basic dynamics (see, for example, [3]), driven by the current motor thrusts $f_{1\cdots4}$:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{J}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}l(f_2 - f_4)\\ l(f_3 - f_1)\\ \kappa(f_1 - f_2 + f_3 - f_4)\end{bmatrix} - \boldsymbol{\omega}\times\mathbf{J}\boldsymbol{\omega}, \tag{A.3}$$

$$c = (f_1 + f_2 + f_3 + f_4)/m,$$

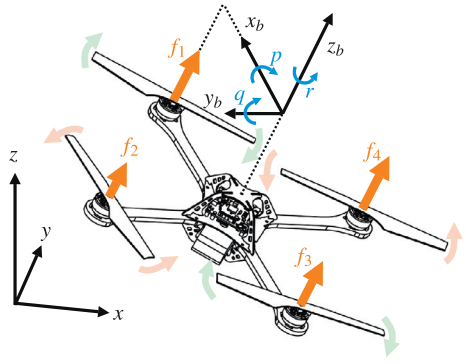

Fig. A.19. Coordinate systems, propeller directions, and motor numbering used in this work.

where $\mathbf{J}$ is the inertia matrix of the vehicle, l is the vehicle center to rotor distance, κ is an experimentally determined constant and m is the mass of the vehicle. We model each motor as a first-order system with constraints $0 < f_{min} \leqslant f_i \leqslant f_{max}$. The motors cannot be reversed or stopped in flight. Experiments show that the motors are quicker to spin up than to spin down:

$$\dot{f}_i = \frac{1}{\tau_f}(\check{f}_i - f_i), \tag{A.4}$$

where $\check{f}_i$ is the desired thrust command from the on-board feedback controller as explained in Appendix B and $\tau_f = \tau_{f,up}$ when $\check{f}_i > f_i$ and $\tau_f = \tau_{f,dn}$ otherwise.

Appendix B. On-board control

The commands send to the vehicle are desired body rates and a desired mass-normalized thrust: $\check{p}, \check{q}, \check{r}$, and $\check{c}$. The on-board control system is tasked with following these commands, using high-rate on-board sensing such as rate gyros in feedback when available.

More concretely, given a desired command and estimated body rates from the gyroscopic sensors, the on-board controller produces individual motor thrust commands, $\check{f}_{1\ldots4}$. With the dynamics to be controlled, Eq. (A.3), being of first order, the control law is chosen in a feedback linearizing loop-shaping approach with the closed-loop dynamics of a first-order linear system. To find the desired motor commands, we invert Eq. (A.3):

$$\begin{bmatrix} l(\check{f}_2 - \check{f}_4) \\ l(\check{f}_3 - \check{f}_1) \\ \kappa(\check{f}_1 - \check{f}_2 + \check{f}_3 - \check{f}_4) \end{bmatrix} = \mathbf{J}\begin{bmatrix} \frac{1}{\tau_{pq}}(\check{p} - p) \\ \frac{1}{\tau_{pq}}(\check{q} - q) \\ \frac{1}{\tau_r}(\check{r} - r) \end{bmatrix} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}, \tag{B.1}$$

$$(\check{f}_1 + \check{f}_2 + \check{f}_3 + \check{f}_4) = m\check{c}.$$

It is then straightforward to solve for the propeller forces $\check{f}_{1\ldots4}$. The closed-loop time constants τ_{pq} and τ_r are tuned to provide fast response, with a lower bound caused by stability requirements in combination with the bandwidth of the motor speed control loop.

A lower control loop then performs speed control for each individual propeller in order to produce the desired propeller forces $\check{f}_{1\ldots4}$. This task is performed by the motor controllers, which are off-the-shelf components [20]. Because access to the programs running on these controllers is not available, we consider this control loop as given and have only identified it, as shown in (A.4).

Appendix C. On-board calibration

The gyroscopes are initially calibrated for bias when the vehicle is switched on. This initial calibration is then augmented by a more comprehensive calibration routine, typically conducted in preparation for an experiment after first takeoff. Here the vehicle is commanded to a hover, from which point we can estimate gyro biases, propeller efficiency factors and a measurement frame misalignment.

The gyro biases are taken as the mean value of the gyro output, since the mean angular velocity is known to be zero. Next we define "propeller factors" $\gamma_{1\ldots4}$, as the ratio between the true thrust $f_{1\ldots4}$ and the nominally commanded thrust $\check{f}_{1\ldots4}$, i.e. $f_{1\ldots4} = \gamma_{1\ldots4}\check{f}_{1\ldots4}$. Taking (A.3) and setting the angular rates to zero and the collective acceleration to gravity ($c = g$), we obtain a simple matrix equation to solve for $\gamma_{1\ldots4}$,

$$\begin{bmatrix} 0 & l\check{f}_2 & 0 & -l\check{f}_4 \\ -l\check{f}_1 & 0 & l\check{f}_3 & 0 \\ \kappa\check{f}_1 & -\kappa\check{f}_2 & \kappa\check{f}_3 & -\kappa\check{f}_4 \\ \frac{\check{f}_1}{m} & \frac{\check{f}_2}{m} & \frac{\check{f}_3}{m} & \frac{\check{f}_4}{m} \end{bmatrix}\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g \end{bmatrix}. \tag{C.1}$$

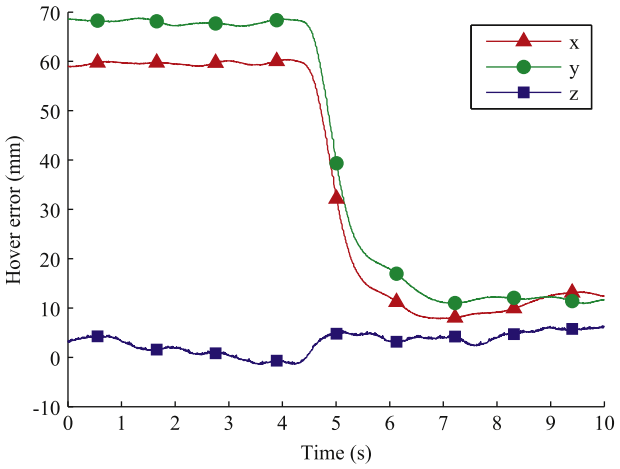

Fig. C.20. Effects of static calibration on hover accuracy. The mean hover error is reduced from approximately 89–16 mm after the static calibration routine, which terminates at 5 s. Note that no form of integral control is used.

These factors are then used to scale all subsequent inputs to the motor, and are a useful diagnostic tool: if the calibration routine results in significantly differing factors, this is a strong indication of damage to the motors/propellers.

Attached to each vehicle is a set of at least three uniquely configured markers observed by the motion capture system. These markers define a measurement frame, which is approximately aligned with the body fixed frame. The pitch and roll components of this misalignment are estimated during the static calibration, as the z_b-axis of the body fixed frame must align with gravity during hover.

The calibration procedure significantly improves the tracking performance of the vehicles: representative results are shown in Fig. C.20, where the mean hover error is reduced to less than 20 mm. The propeller factors measured were $\{1.02, 1.25, 0.84, 0.95\}$ respectively for the front, left, rear and right propellers. The calibration also showed a measurement frame misalignment of 0.6°, and gyro biases of (0.05, −0.02, −0.01) rad/s around the (x_b, y_b, z_b) axes.

Appendix D. Trajectory tracking controller

An overview of the cascaded controllers used for flying a quadrocopter is shown in Fig. C.21. This control scheme is most commonly used to follow desired vehicle position and yaw trajectories.

The off board control consists of four separate loops: horizontal and vertical position control loops, a reduced attitude control loop and a yaw control loop. The output of the four control loops are the three body rate commands to the vehicle, and the collective thrust command.

The estimation and latency compensation presented in Section 3.2 allows us to do full state feedback controllers with no consideration of latency. While the off-board controllers are executed in discrete time to generate commands at the rate of vehicle-ground communication (see Section 3.4), they are designed based on the continuous-time system dynamics representation with no consideration of the sampling rate. Because the communications channel is significantly faster than the system dynamics, performance losses from this are considered to be minor.

D.1. Vertical control

The vertical control loop is shaped such that it responds to altitude errors like a second-order system with a time constant τ_z and damping ratio ζ_z:

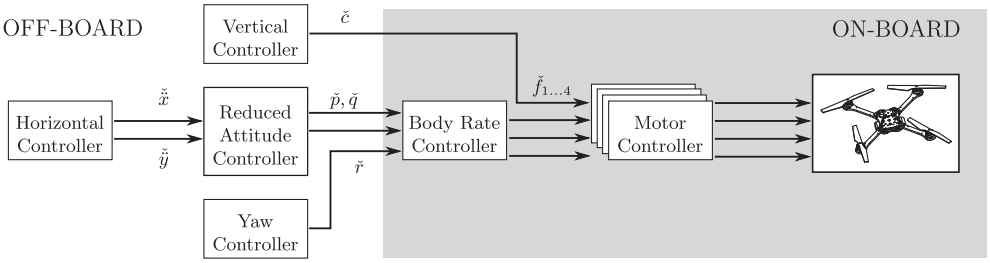

Fig. C.21. Overview of the quadrocopter cascaded control loops. Feedback signal flow is omitted for clarity.

$$\ddot{z} = \frac{1}{\tau_z^2}(\check{z} - z) + \frac{2\zeta_z}{\tau_z}(\dot{\check{z}} - \dot{z}) + \ddot{\check{z}}, \tag{D.1}$$

where $\check{z}, \dot{\check{z}}$, and $\ddot{\check{z}}$ are the desired vertical position, velocity, and acceleration, respectively.

Using the translational system dynamics (A.1) and the desired dynamics, the collective thrust is found to be

$$\check{c} = \frac{1}{\mathbf{R}_{33}}\left(\frac{1}{\tau_z^2}(\check{z} - z) + \frac{2\zeta_z}{\tau_z}(\dot{\check{z}} - \dot{z}) + \ddot{\check{z}} + g\right), \tag{D.2}$$

where the scalar $\mathbf{R}_{33}$ is the (3,3) element of the vehicle rotation matrix $\mathbf{R}$.

D.2. Horizontal control

Similarly to the vertical control loop, the two horizontal control loops are shaped to behave in the manner of a second-order system, with time constant τ_{xy} and damping ratio ζ_{xy}. However, no control inputs are directly calculated. Instead, the commanded accelerations $\ddot{\check{x}}$, $\ddot{\check{y}}$ are given as set points to the attitude controller.

D.3. Reduced attitude control

The attitude controller controls the reduced attitude [32] of the vehicle such that the commanded accelerations $\ddot{\check{x}}$, $\ddot{\check{y}}$ are met. Rewriting the translational dynamics Eq. (A.1), it can be seen that two matrix elements determine the translational acceleration, together with the collective thrust command:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{13} \\ \mathbf{R}_{23} \end{bmatrix} \check{c}. \tag{D.3}$$

The commanded accelerations are converted to commanded rotation matrix entries using the above relationship. To avoid the vehicle losing altitude when large horizontal accelerations are commanded, the rotation matrix entries are limited when the collective thrust command $\check{c}$ reaches the maximum allowable value. The attitude control loop is shaped such that the two matrix entries react in the manner of a first order system with time constant τ_{rp}. Using the rotational kinematics Eq. (A.2), the rate of change of the matrix entries can be used to compute the desired vehicle body rates $\check{p}$ and $\check{q}$:

$$\begin{bmatrix} \check{p} \\ \check{q} \end{bmatrix} = \frac{1}{\mathbf{R}_{33}} \begin{bmatrix} \mathbf{R}_{21} & -\mathbf{R}_{11} \\ \mathbf{R}_{22} & -\mathbf{R}_{12} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{R}}_{13} \\ \dot{\mathbf{R}}_{23} \end{bmatrix}. \tag{D.4}$$

D.4. Yaw control loop

The above controllers fully define the translational behavior of the quadrocopter vehicle. There is, however, a remaining degree of freedom: Arbitrary rotations about the z_b-axis of the vehicle do not affect the above dynamics. To control this remaining degree of freedom, we use the Euler angle parametrization [33] of the rotation matrix $\mathbf{R}$, commonly referred to as yaw-pitch-roll, and consisting of a pure rotation about the global z axis (yaw), followed by a rotation about the new y-axis (pitch), and a final rotation about the new x-axis (roll).

The yaw controller is a proportional controller from the measured yaw angle to the yaw angle rate. The yaw angle rate is then mapped to the vehicle body rate r using the kinematic relations of Euler angles.

References

[1] How J, Bethke B, Frank A, Dale D, Vian J. Real-time indoor autonomous vehicle test environment. IEEE Control Syst Mag 2008;28(2):51–64. http://dx.doi.org/10.1109/MCS.2007.914691.

[2] Hoffmann GM, Huang H, Waslander SL, Tomlin CJ. Precision flight control for a multi-vehicle quadrotor helicopter testbed. Control Eng Practice 2011;19(9):1023–36. http://dx.doi.org/10.1016/j.conengprac.2011.04.005.

[3] Michael N, Mellinger D, Lindsey Q, Kumar V. The GRASP multiple micro-UAV testbed. IEEE Robot Autom Mag 2010;17(3):56–65. http://dx.doi.org/10.1109/MRA.2010.937855.

[4] Mellinger D, Michael N, Kumar V. Trajectory generation and control for precise aggressive maneuvers with quadrotors. Int J Robot Res 2012;31(5):664–74. http://dx.doi.org/10.1177/0278364911434236.

[5] Gillula JH, Huang H, Vitus MP, Tomlin CJ. Design and analysis of hybrid systems, with applications to robotic aerial vehicles. In: 2009 International symposium of robotics research; 2009.

[6] Lupashin S, D'Andrea R. Adaptive fast open-loop maneuvers for quadrocopters. Auton Robots 2012;33:89–102. http://dx.doi.org/10.1007/s10514-012-9289-9.

[7] Hehn M, D'Andrea R. A flying inverted pendulum. In: 2011 IEEE international conference on robotics and automation (ICRA); 2011. p. 763–70. doi: 10.1109/ICRA.2011.5980244.

[8] Ritz R, Müller M, Hehn M, D'Andrea R. Cooperative quadrocopter ball throwing and catching. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS); 2012. p. 4972–8. doi: 10.1109/IROS.2012.6385963.

[9] Lindsey Q, Mellinger D, Kumar V. Construction of cubic structures with quadrotor teams. In: Robotics: science and systems; 2011.

[10] Saatchi & Saatchi new directors' showcase at Cannes Lions; 2012. <http://www.canneslions.com/saatchinewdirectors/>.

[11] Ars Electronica 2012: Spaxels; 2012. <http://www.aec.at/quadcopter/en>.

[12] Trimpe S, D'Andrea R. Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom. In: 2010 IEEE international conference on robotics and automation (ICRA); 2010. p. 2630–6. doi: 10.1109/ROBOT.2010.5509756.

[13] Oung R, DAndrea R. The distributed flight array. Mechatronics 2011;21(6):908–17. http://dx.doi.org/10.1016/j.mechatronics.2010.08.003.

[14] Achtelik M, Achtelik M, Weiss S, Siegwart R. Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments, in: 2011 IEEE international conference on robotics and automation (ICRA); 2011. p. 3056–63. doi: 10.1109/ICRA.2011.5980343.

[15] Smart B. Competitions, challenges, or journal papers? [competitions]. IEEE Robot Autom Mag 2012;19(1):14. http://dx.doi.org/10.1109/MRA.2012.2186709.

[16] D'Andrea R, Babish M. The RoboFlag testbed. In: Proceedings of the American control conference, vol. 1; 2003. p. 656–60. doi: 10.1109/ACC.2003.1239094.

[17] Veloso M, Stone P, Han K. CMUnited-97: RoboCup-97 small-robot world champion team. AI Mag 1998;19(3):61–9.

[18] Stubbs A, Vladimerou V, Fulford A, King D, Strick J, Dullerud G. Multivehicle systems control over networks: a hovercraft testbed for networked and decentralized control. IEEE Control Syst 2006;26(3):56–69. http://dx.doi.org/10.1109/MCS.2006.1636310.

[19] Miller I, Campbell M, Huttenlocher D, Nathan A, Kline F-R, Moran P, et al. Team Cornells Skynet: robust perception and planning in an urban environment. In: Buehler M, Iagnemma K, Singh S, editors. The DARPA urban challenge. Springer tracts in advanced robotics, vol. 56. Berlin Heidelberg: Springer; 2009. p. 257–304. http://dx.doi.org/10.1007/978-3-642-03991-1_7.

[20] Gurdan D, Stumpf J, Achtelik M, Doth K-M, Hirzinger G, Rus D. Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz. In: 2007 IEEE international conference on robotics and automation; 2007. p. 361–6. doi: 10.1109/ROBOT.2007.363813.

[21] Sherback M, Purwin O, D'Andrea R. Real-time motion planning and control in the 2005 Cornell RoboCup system. In: Kozlowski K, editor. Robot motion and control. Lecture notes in control and information sciences, vol. 335. Berlin/Heidelberg: Springer; 2006. p. 245–63. http://dx.doi.org/10.1007/978-1-84628-405-2_16.
[22] Mueller M, D'Andrea R. Critical subsystem failure mitigation in an indoor uav testbed. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS); 2012. p. 780–5. doi: 10.1109/IROS.2012.6385910.
[23] Plett GL. Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs part 1: background. J Power Sour 2004;134:252–61.
[24] Müller M, Lupashin S, D'Andrea R. Quadrocopter ball juggling. In: 2011 IEEE/RSJ international conference on intelligent robots and systems (IROS); 2011. p. 5113–20. doi: 10.1109/IROS.2011.6094506.
[25] Mueller F, Schoellig A, D'Andrea R. Iterative learning of feed-forward corrections for high-performance tracking. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS); 2012. p. 3276–81. doi: 10.1109/IROS.2012.6385647.
[26] Ambühl A. Human interaction with a quadrocopter using a Kinect sensor. Bachelor thesis, ETH Zürich; 2011.
[27] Augugliaro F, Schoellig A, D'Andrea R. Generation of collision-free trajectories for a quadrocopter fleet: a sequential convex programming approach. In: 2012 IEEE/RSJ international conference on intelligent robots and systems (IROS); 2012. p. 1917–22. doi: 10.1109/IROS.2012.6385823.
[28] Schoellig AP, Augugliaro F, D'Andrea R. A platform for dance performances with multiple quadrocopters. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS) – workshop on robots and musical expressions; 2010. p. 1–8.
[29] Brescianini D, Hehn M, D'Andrea R. Quadrocopter pole acrobatics. In: Proc. of the IEEE/RSJ international conference on intelligent robots and systems (IROS); 2013.
[30] Augugliaro F, D'Andrea R. Admittance control for physical human–quadrocopter interaction. In: European control conference; 2013.
[31] Hughes PC. Spacecraft attitude dynamics. John Wiley & Sons; 1986.
[32] Chaturvedi N, Sanyal A, McClamroch N. Rigid-body attitude control. IEEE Control Syst 2011;31(3):30–51. http://dx.doi.org/10.1109/MCS.2011.940459.
[33] Zipfel PH. Modeling and simulation of aerospace vehicle dynamics. Second ed. AIAA; 2007.