Build Your Own Visual-Inertial Drone

A Cost-Effective and Open-Source Autonomous Drone

By Inkyu Sa, Mina Kamel, Michael Burri, Michael Bloesch, Raghav Khanna, Marija Popović, Juan Nieto, and Roland Siegwart

This article describes an approach to building a cost-effective and research-grade visual-inertial (VI) odometry-aided vertical takeoff and landing (VTOL) platform. We utilize an off-the-shelf VI sensor, an onboard computer, and a quadrotor platform, all of which are factory calibrated and mass produced, thereby sharing similar hardware and sensor specifications [e.g., mass, dimensions, intrinsic and extrinsic of camera–inertial measurement unit (IMU) systems, and signal-to-noise ratio]. We then perform system calibration and identification, enabling the use of our VI odometry, multisensor fusion (MSF), and model predictive control (MPC) frameworks with off-the-shelf products. This approach partially circumvents the tedious parameter-tuning procedures required to build a full system. The complete system is extensively evaluated both indoors using a motion-capture system and outdoors using a laser tracker while performing hover and step responses and trajectory-following tasks in the presence of external wind disturbances. We achieve root-mean-square (rms) pose errors of 0.036 m with respect to reference hover trajectories. We also conduct relatively long distance (≈180 m) experiments on a farm site, demonstrating a 0.82% drift error of the total flight distance. This article conveys the insights we acquired about the platform and sensor module and offers open-source code with tutorial documentation to the community.

Digital Object Identifier 10.1109/MRA.2017.2771326
Date of publication: 21 December 2017

Overview of VTOL Microaerial Vehicles

During the past decade, VTOL microaerial vehicles (MAVs), which use counterrotating rotors to generate thrusting and rotational forces, have gained renown in both research and industry. Emerging applications, including structural inspection, aerial photography, cinematography, and environmental surveillance, have spurred a wide range of ready-to-fly

1070-9932/17©2017IEEE

commercial platforms whose performance has steadily improved in terms of flight time, payload, and safety-related smart features, allowing for more stable, easier, and safer pilot maneuvers. However, a key challenge is directly adapting commercial platforms [1] for tasks requiring accurate dynamic models, high-performance controllers, and precise, low-latency state estimators, such as obstacle avoidance and path planning [2]–[5], landing on moving platforms [6], object picking [7], and precision agriculture [8].

Research-grade MAV platforms can alleviate these issues. For instance, Ascending Technologies of Krailling, Germany, provides excellent products [9], [10] for advanced aerial applications [11] with an accompanying software development kit (SDK), scientific resources (including self-contained documentation), and online support. However, they are relatively expensive for research purposes, and part replacements are difficult due to limited retail services.

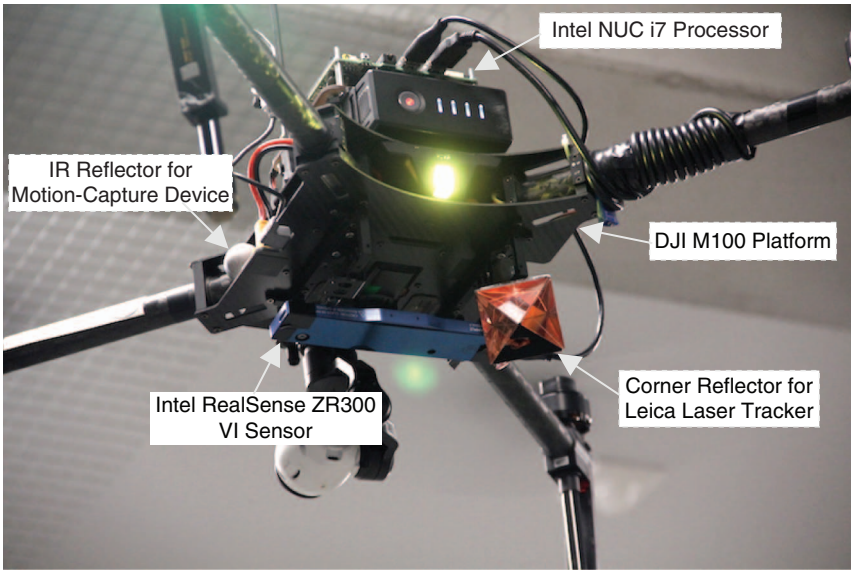

Figure 1. A Matrice 100 VTOL MAV quadrotor platform with a downward-facing VI sensor. The infrared (IR) reflectors and prism are mounted for obtaining the ground truth.

For VTOL MAVs to become more widespread, they will require lower costs and more easily obtainable parts. Recently, the DJI Matrice 100 MAV (Shenzhen, China, https://www.dji.com/matrice100) (Figure 1) has been introduced as a commercial platform with parts available at local retailers. Developers can access sensor data, e.g., from the IMU and barometers and send commands to the low-level attitude controller through the SDK [12]. Although manufacturer documentation is provided, there are limited scientific resources detailing aspects like attitude dynamics, the underlying autopilot controller structure, and input command scaling. This information is critical for subsequent position control strategies, such as MPC.

A VI sensor is a favorable choice for aerial robotics due to its light weight, low power consumption, and ability to recover unknown scales (monocular camera). This sensor suite can provide a time-synchronized wide field of view (FoV) image (≈133°) and motion measurements. There is an impressive research-grade VI sensor [13] providing time-synchronized and calibrated IMU–stereo camera images with supporting documentation and device drivers. Unfortunately, this sensor is relatively expensive, and its production was discontinued. The Intel ZR300 (http://click.intel.com/realsense.html) has recently emerged as a compelling alternative. It is an affordable and mass-produced module that enables applying the same configuration and calibration parameters to other sensors with low performance penalties. Table 1 summarizes the total cost for the VI sensor, MAV, and personal computer (PC).

In this article, we address these gaps by using a ready-to-market affordable VTOL platform and a VI sensor to perform system identification, sensor calibration, and system integration. The system can autonomously track motion commands with high precision in indoor and outdoor environments. This ability is fundamental for any high-level application, enabling researchers to build custom systems without tedious tuning procedures. Moreover, we provide self-contained documentation to cater for setups with different inertial moments and sensor mount configurations. The contributions of this system article are

- a delivery of software packages [including a modified SDK, nonlinear MPC (nMPC), calibration parameters, and system identification tools] with accompanying documentation to the community
- an evaluation of the control and state estimation performance in different environments

Table 1. A summary of total system cost.

Identification	Name	Price (US$)
1	M100*	1,979.40
2	VI sensor	289
3	Intel NUC i7	482
4	1-TB solid-state drive	280
5	16-GB RAM	103.5
6	Transistor–transistor logic to USB converter (Future Technology Devices International, Glasgow, Scotland)	14.75
7	Power regulator (12 V)	21
Total		**3,169.65**

*You can get a 40% discount with the developer registration (https://developer.dji.com/). Since the registration is mandatory to send commands to the N1 autopilot, we list the 40% discounted price in the parts list.

- a demonstration of use cases adapting our approach to build custom research platforms with a step-by-step tutorial available at https://goo.gl/yj8WsZ.

The techniques presented are independent from the platforms used in this article, e.g., the MPC strategy is applicable to other commercial VTOL platforms (such as the Matrice 200 or 600) given identified dynamic system models using our procedures. The robust VI odometry (ROVIO) framework can also be utilized in mobile platforms for navigation tasks [14] with the VI sensor (Intel ZR300).

Related Work

VTOL MAVs are increasingly used for tasks such as building inspections, aerial photography, and precision agriculture. To perform these missions, capabilities of localization, perception, control, and path planning are critical. These topics are very broad, and it is challenging to cover them comprehensively in this article. We thus focus on state-of-the-art VI odometry techniques, dynamic system identification, and VTOL MAV control as the most relevant subtopics.

VI odometry has been an active research topic in the last decade, given advances in microelectromechanical systems, IMUs, and imaging technologies [15]. It is a favorable option for payload-constrained platforms like VTOL MAVs due to its light weight and relatively low computational requirements. All sensor states are jointly coupled (i.e., tightly coupled) within 1) filtering or 2) nonlinear optimization frameworks. These frameworks estimate a robot's ego-motion by integrating visual and inertial measurements while minimizing data preprocessing and, thereby, improving stochastic consistency. Filter-based approaches often do this within a Kalman filter (e.g., [16]), thus the approach suffers from drift and exhibits only a limited representation of the global environment. However, the resulting localization accuracy suffices for stabilizing control and executing local tasks. Furthermore, these frameworks procure estimates that can be input to feedback control strategies at a constant computational cost. In contrast, the second class of methods performs key frame-based nonlinear optimization over all jointly coupled sensor states [17]–[21]. In some cases, these approaches can also perform loop closures to compensate for drifting behavior and provide globally consistent maps. However, they often demand costly computations that may burden smaller platforms, and they demonstrate state estimations without feedback control (or only for hovering tasks).

Identifying the dynamics of attitude controllers is vital to achieving sufficient control performance. For a common quadrotor, the rigid vehicle dynamics are well known [22], [23] and can be modeled as a nonlinear system with individual rotors attached to a rigid airframe, accounting for drag force and blade flapping [24]. However, the identification of attitude controllers is often nontrivial for consumer products due to limited scientific resources and, thus, requires techniques to estimate dynamic model parameters. Traditionally, parameter estimation is performed offline using complete measurement data obtained from a physical test bed and computer-aided design models [25], [26]. A linear least-squares method is used to estimate parameters from recorded data in batch offline processing [27], [28]. This approach only requires flight data sets; however, identification must be repeated if the vehicle configuration changes. In contrast, online system identification involves applying recursive estimation to real-time sensor data. Burri et al. [29] present a method for identifying the dominant dynamic parameters of a VTOL MAV using the maximum likelihood approach, and they apply it to estimate moments of inertia and aerodynamic parameters. We follow a batch-based strategy to determine the dynamic vehicle parameters from short manual-piloting maneuvers. This allows us to obtain the parameters needed for MPC [30] using only the onboard IMU and without restrictive simplifying assumptions.

VTOL MAV Platform and VI Sensor Module

This section describes the vehicle (Matrice 100) and sensor module (Intel ZR300) used in this article. Their general specifications are well documented and available from official sources. Here, we only highlight the information relevant for building research platforms, including autotrim compensation, dead-zone recovery, camera–IMU extrinsic calibration, and their time synchronization.

VTOL MAV Platform

Our vehicle is a quadrotor with a diagonal length of 650 mm and four 13-in diameter propellers with a 4.5-in thread pitch. The maximum takeoff weight is 3,600 g, and flight time varies depending on the hardware configuration (16–40 min). The N1 autopilot manages attitude control and telemetry, but information regarding the device is not publicly disclosed. Using the SDK, sensor data can be accessed through serial communication, and we configure the IMU update rate at 50 Hz. The SDK enables access to most functionalities and supports cross-platform development environments, such as the Robot Operating System (ROS), Android, and iPhone operating system. However, there is a fundamental issue in sending control commands with this protocol. The manufacturer uses `ROS services` to send commands; this is not recommended (http://wiki.ros.org/ROS/Patterns/Communication), as they are blocking calls that should be used only for triggering signals or quick calculations. If a data transaction (handshaking) fails (e.g., due to poor Wi-Fi signal), then it blocks all subsequent calls. Given that low control command latency (≈20 ms) can have large performance impacts, we modify the SDK to send direct control commands through the serial port. We faced communication problems (921,600 b/s) while receiving/sending data at 100 Hz, as the onboard computer could not transmit any commands to the N1 autopilot at this particular frequency. Consequently, we used 50 Hz in all presented experiments and are currently investigating this issue.

There are usually trim switches on an ordinary transmitter that allow for small command-input adjustments. The N1 autopilot, however, has an autotrim feature that balances attitude by

estimating horizontal velocity. This permits easier and safer manual piloting but introduces a constant position error offset for autonomous control. To address this, we estimate the balancing point where the vehicle's motion is minimum (hovering) and adjust the neutral position to this point. If there is a change in an inertial moment (e.g., mounting a new device or changing the battery position), the balancing position must be updated.

Another interesting autopilot feature is the presence of a dead zone in the small range close to the neutral value, where it ignores all input commands. This function is also useful for manual piloting because the vehicle should ignore perturbations from hand tremors, but it significantly degrades control performance. We determine this range by sweeping control commands around the neutral stick position of a transmitter and detecting the control inputs when any motion is detected (i.e., horizontal and vertical velocity changes). As this task is difficult with a real VTOL platform due to its fast and naturally unstable dynamics, we use the hardware-in-loop simulator, enabling input command reception from the transmitter. If the commands are within those ranges, then they are set as the maximum/minimum dead zone values.

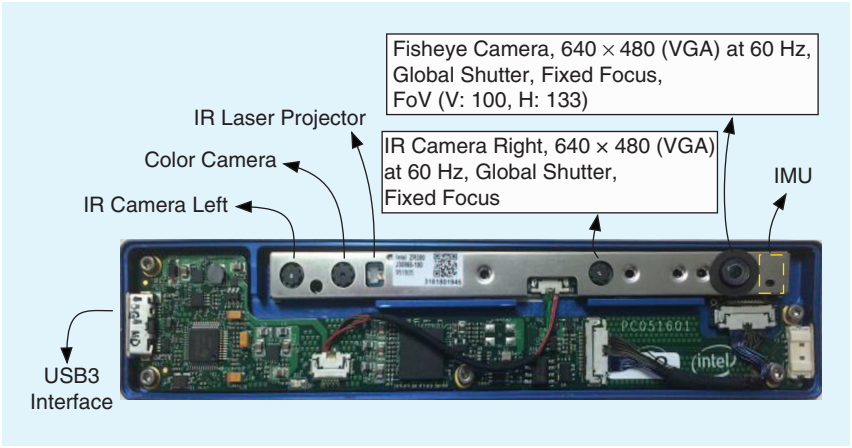

Figure 2. An Intel ZR300 VI sensor module [38]. VGA: video graphics array. USB: universal serial bus. (Image courtesy of http://goo.gl/DL93Wo.)

VI Sensor

In this article, we exploit a ready-to-market VI sensor [37]. Most importantly, the sensor has one fisheye camera with an FoV of 133° and 100° in the horizontal and vertical directions, respectively, and streams a 640 × 480 image at 60 frames/s, as shown in Figure 2. An onboard IMU provides three-axis accelerations and angular velocities in a body frame with time stamps at 20 kHz. As we do not use depth measurements, the two IR cameras; projector; and red, green, and blue camera are disabled for reduced-power operation. Camera–IMU time synchronization for any VI-related task is nontrivial because camera images and motions measured by the IMU are tightly connected.

Camera–IMU Time Synchronization

The ZR300 has different clock sources for the IMU and image time stamps. Therefore, direct usage of the time stamps from the RealSense library leads to a poor estimator performance. To mitigate this issue, a synchronization message is generated every time the sensor captures an image. This message contains the time stamp and sequence number of the corresponding image, and the same sequence number is also contained in the image message. We implemented two ring buffers for images and synchronization messages to look up the correct time stamp of the image with respect to the IMU before publishing it over the ROS. This procedure is depicted in Figure 3.

Another important aspect is that the sensor IMU has different sampling rates on its gyroscopes (~200 Hz) and accelerometers (~250 Hz). Because the noise density of the gyroscopes is smaller than that of the accelerometers and the data is more important for state estimation, we publish an IMU message containing both sensors at the rate of the gyroscopes. This requires buffering the messages and linear interpolation of the accelerometer messages. In our present

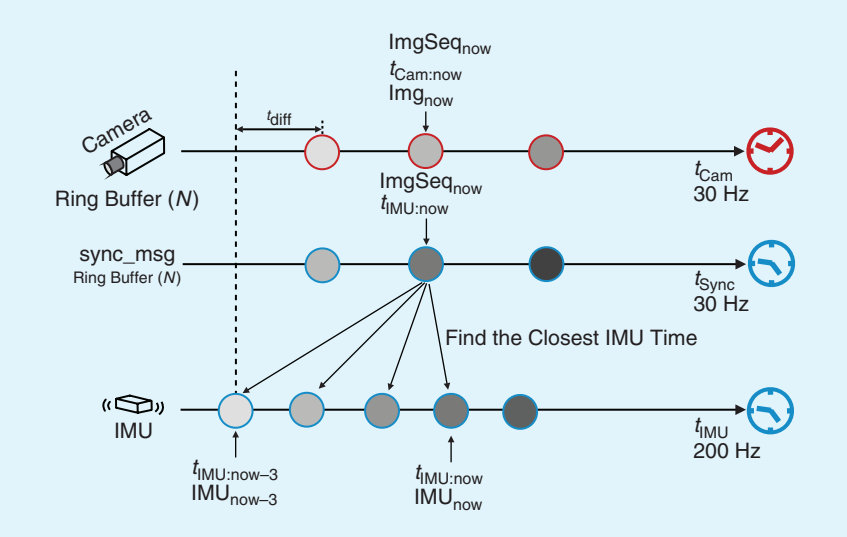

Figure 3. A camera–IMU time-synchronization illustration. Two sensors are running at different rates with their own time sources as indicated by the red and blue clocks. The faster update rate of the IMU, $t_{IMU:now}$, is used as the master time. For each node, the shade of gray represents amount of time elapsed (i.e., the same shade signifies the same time). N denotes the size of the ring buffers.

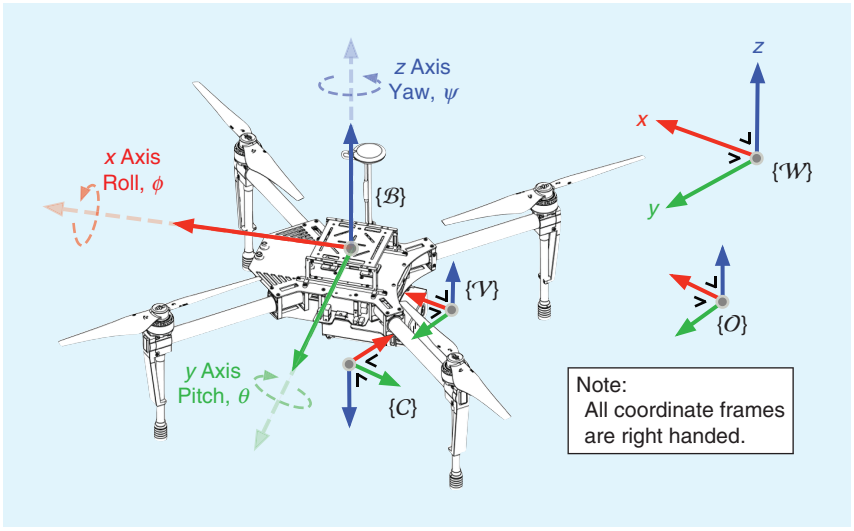

Figure 4. The coordinate system definitions. There are five frames: world $\{\mathcal{W}\}$, odometry $\{\mathcal{O}\}$, body $\{\mathcal{B}\}$, camera $\{\mathcal{C}\}$, and VI sensor IMU $\{\mathcal{V}\}$. (Image courtesy of http://goo.gl/7NsbmG.)

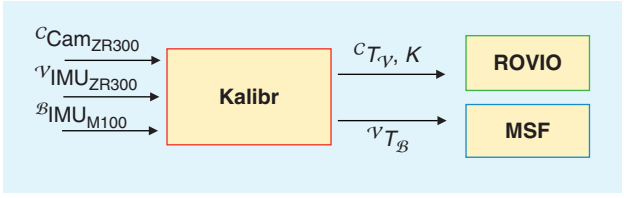

Figure 5. The three inputs of image and IMUs are fed into the Kalibr calibration framework. Intrinsic (K) and extrinsic ($^{\mathcal{C}}T_{\mathcal{V}}$ and $^{\mathcal{V}}T_{\mathcal{B}}$) are utilized by subsequent VI odometry and sensor fusion frameworks.

version of the sensor, the IMU is not intrinsically calibrated. We use the extended version of Kalibr [31] to estimate each axis of the gyro and the accelerometer with respect to a reference. Finally, we currently do not compensate for the camera exposure time. The time stamps are triggered at the beginning of the exposure time rather than in the middle. This could be important in cases of large lighting changes. We use a constant offset between the IMU and camera as estimated using Kalibr [32].

Coordinate Systems Definition

We define five right-handed frames following standard ROS conventions: world $\{\mathcal{W}\}$, odometry $\{\mathcal{O}\}$, body $\{\mathcal{B}\}$, camera $\{\mathcal{C}\}$, and VI sensor IMU $\{\mathcal{V}\}$, as shown in Figure 4. $\{\mathcal{B}\}$ is aligned with the vehicle's IMU frame, and its x axis indicates the forward direction of the vehicle, with the y and z axes as left and up, respectively. We use Euler angles; roll ϕ, pitch θ, and yaw ψ about the x, y, and z axis, respectively, for the rms error calculation and visualization. Quaternions are utilized for any computational processes. Note that the default vehicle coordinate system configuration is North-East-Down so that the angle, acceleration, and angular velocity measurements from the onboard IMU are rotated π along the x axis to align with $\{\mathcal{B}\}$. The $\{\mathcal{W}\}$ and $\{\mathcal{O}\}$ frames are fixed coordinates where VI odometry is initialized. We treat them identically in this article, but they can differ if external pose measurements (e.g., global positioning system or Leica laser tracker) are employed. These coordinate systems and notations are used in the rest of this article.

Extrinsic Calibration

Two essential transformations are required before a flight: 1) the transformation from the vehicle IMU frame $\{\mathcal{B}\}$ to the VI sensor's camera frame $\{\mathcal{C}\}$, i.e., $^{\mathcal{B}}\boldsymbol{T}_{\mathcal{C}} \in SE(3) \subset \mathbb{R}^{4\times4}$, and 2) the transformation from the camera frame $\{\mathcal{C}\}$ to VI sensor's IMU frame $\{\mathcal{V}\}$, $^{\mathcal{C}}\boldsymbol{T}_{\mathcal{V}}$. These extrinsic parameters and camera intrinsics are identified with Kalibr [32]. ROVIO and MSF employ this calibration data, as shown in Figure 5. $^{\mathcal{C}}\text{Cam}_{\text{ZR300}}$, $^{\mathcal{V}}\text{IMU}_{\text{ZR300}}$, and $^{\mathcal{B}}\text{IMU}_{\text{M100}}$ denote an image in $\{\mathcal{C}\}$ frame taken by the VI sensor, IMU message in $\{\mathcal{V}\}$ from the VI sensor, and IMU message in $\{\mathcal{B}\}$ from Matrice 100's autopilot, respectively. $\boldsymbol{K}$ is an intrinsic camera parameter describing a lens surface using an equidistant distortion model. Note that this procedure only needs to be performed once if the VI sensor configuration (e.g., position and orientation) changes. Otherwise, predefined parameters can be loaded without the calibration process.

VI Odometry Framework

Highlights of our ROVIO framework [16] approach are threefold:

1) It directly leverages pixel intensity errors as an innovation term inside the extended Kalman filter (EKF), thereby resulting in a tighter fusion of visual and inertial sensor data. The framework employs multilevel image patches as landmark descriptors (see Figure 6); therefore, the computationally expensive visual feature descriptor extraction step can be avoided.
2) It makes use of full robocentric formulation to avoid any possible corruption of unobservable states, and, therefore, the consistency of the estimates can be improved. The landmark locations are parameterized by bearing vector and distance parameters with respect to the current camera pose to improve modeling accuracy. This is particularly beneficial for fast landmark initialization and circumvents a complicated and cumbersome initialization procedure. The bearing vectors are represented as members of the two-dimensional manifold $\mathcal{S}^2$, and minimal differences/derivatives are employed for improved consistency and efficiency (the filter can be run on a single standard central processing unit core).
3) The IMU biases and camera–IMU extrinsics are also included in the filter state and coestimated online for higher accuracy.

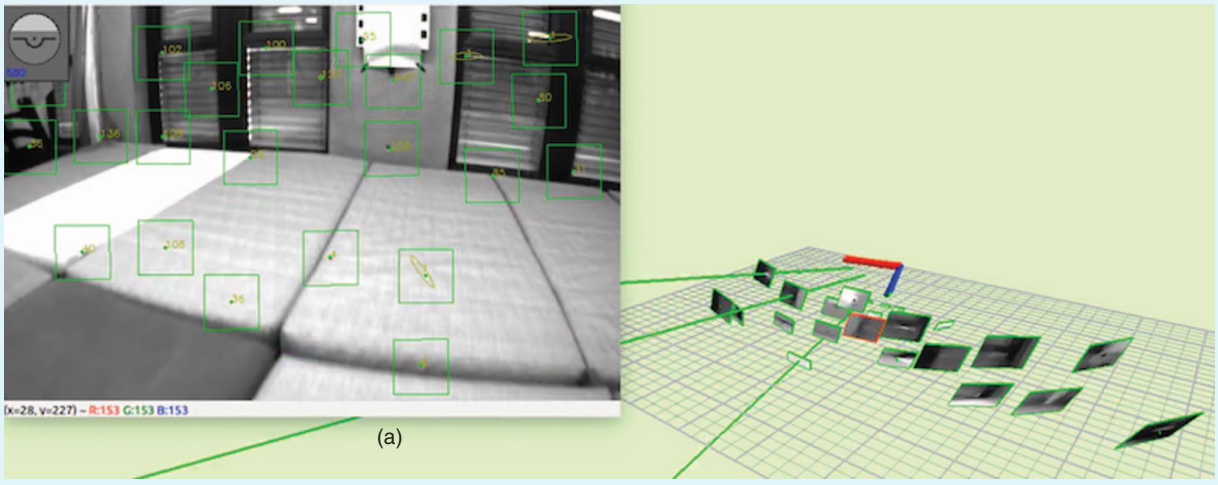

Figure 6. A snapshot of ROVIO running with a MAV data set [33]. (a) The figure shows tracked landmarks (green dots on the center of green squares and their corresponding identification). The squares are projected patches that are used for building up multilevel patches from an image pyramid. The predicted landmark location uncertainties are represented by yellow ellipses. (b) The figure shows 3-D estimated camera poses and tracked patches, including their distance uncertainties [39]. (Image courtesy of https://goo.gl/q5zCRV.)

Figure 6 illustrates an instance of feature tracking and pose estimation with ROVIO. First, a large number of key point candidates are extracted using a feature from an accelerated segment test corner detector. Given the current tracked feature set, candidates are filtered out if they are close to the current features based on a threshold (L2-norm distance). After their removal, the adapted Shi–Tomasi score accounting for the combined Hessian on multiple scales is computed for each candidate. A higher Shi–Tomasi score implies better candidate alignment to the corresponding multilevel patch feature. A feature bucketing technique ensures a good distribution of the candidates over the extracted image frame. An adaptive threshold technique is utilized to regulate the total number of features (25 in this article) tracked within an EKF based on local (only last a couple of frames) and global (the latest score because it was detected last time) feature scores.

It is worth noting that ROVIO requires proper parameter tuning for an optimized performance (e.g., the number of features to track, initial covariances, and inverse depth). Although these parameters must be fine-tuned for specific environments, we provide pretuned parameters for the camera–IMU (Intel ZR300) and the MAV (Matrice 100) in office-like indoor and farm-site environments. For EKF modeling and technical details, we refer the reader to our previous work on the ROVIO framework in [16], which includes the filter setup, process and measurement models, and their update procedures.

Dynamic Systems Identification and nMPC

In this section, we summarize our recent work [34] of MAV dynamic systems identification and describe nMPC.

Dynamic Systems Identification

Our nMPC controller requires first-order attitude (roll and pitch) and thrust dynamics for position control and second-order dynamics for the disturbance observer. To identify these dynamic systems, we record input and output data from manual flights. The inputs are the transmitter commands, roll angle (u_ϕ in rad), pitch angle (u_θ in rad), yaw rate (u_ψ in rad/s), and thrust (u_z in N). The output corresponds to the MAV position, orientation, and linear and angular velocities as provided by a motion-capture system. These signals are logged on an onboard computer.

After the data collection, we estimate the scale that maps unitless transmitter input commands (e.g., −1,024~1,024 for pitch command) to the MAV attitude response. This can be determined by linearly mapping with the maximum/minimum angles (±30°) given maximum/minimum input commands; however, in practice, there can be small errors due to, e.g., an unbalanced platform and subtle dynamic differences. Therefore, these scaling parameters are determined using nonlinear least-squares optimization. After this process, we use classic system identification techniques given input and output without time delay estimation option, and estimated dynamic systems are presented in [34].

nMPC for Full Control

The MAV controller is based on nMPC [30]. A cascade approach is used where a high-level nMPC generates attitude commands for a low-level controller running on autopilot. The dynamic behavior of the attitude controller is considered as a first-order system by the high-level controller. The state vector is defined as $\underline{x} = (\underline{p}^T, \underline{v}^T, \phi, \theta, \psi)^T$, where $\underline{p}$ and $\underline{v}$ are the MAV position and velocity, respectively, expressed in the

Table 2. The control performance (rms error) summary.

	Hovering			Step Response			Trajectory Following			
	Indoors		Outdoors	Indoors		Outdoors	Indoors		Outdoors	Unit
Pose	**0.036**	0.049	0.045	0.233	0.395	0.260	0.083	0.091	0.100	m
x	**0.016**	0.022	0.030	0.155	0.277	0.189	0.066	0.042	0.079	m
y	0.018	**0.012**	0.026	0.125	0.230	0.172	0.039	0.071	0.056	m
z	0.026	0.038	0.022	0.122	0.163	0.049	0.030	0.038	**0.024**	m
Roll	**0.863**	1.389	–	–	–	–	1.396	1.593	–	°
Pitch	**0.793**	0.913	–	–	–	–	0.871	1.067	–	°
Yaw	1.573	3.024	–	3.659	6.865	–	**1.344**	2.858	–	°
Duration	30–230	50–180	20–150	30–200	20–120	20–120	30–80	25–120	50–180	s
Wind	–	11–11.5	3.6–7.4	–	11–11.5	3.6–7.4	–	11–11.5	3.6–7.4	m/s

Bold indicates the lowest rms error of each row.

inertial frame, $\{\mathcal{W}\}$. We define the control input vector as $\underline{u} = (u_\phi, u_\theta, u_T)^T$, the reference state at time t, $\underline{x}^{\text{ref}}(t)$, and the steady-state control input at time t, $\underline{u}^{\text{ref}}(t)$. Every time step, the following optimization problem is solved:

$$\begin{aligned}\min_{\underline{U}} \int_{t=0}^{T} &(\underline{x}(t) - \underline{x}^{\text{ref}}(t))^T \underline{Q}_x (\underline{x}(t) - \underline{x}^{\text{ref}}(t)) + (\underline{u}(t) - \underline{u}^{\text{ref}}(t))^T \\ &\underline{R}_u (\underline{u}(t) - \underline{u}^{\text{ref}}(t))\, dt + (\underline{x}(T) - \underline{x}^{\text{ref}}(T))^T \underline{P} (\underline{x}(T) - \underline{x}^{\text{ref}}(T)), \\ \text{subject to} \quad & \dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u}); \\ & \underline{u}(t) \in \mathcal{U}_C, \\ & \underline{x}(0) = \underline{x}(t_0), \end{aligned} \tag{1}$$

where $\underline{Q}_x \succeq 0$ is the penalty on the state error, $\underline{R}_u \succ 0$ is the penalty on control input error, and $\underline{P}$ is the terminal state error penalty. The $\succeq$ operator denotes the positive definiteness of a matrix. The ordinary differential equation representing the MAV dynamics is $\underline{f}(\underline{x}, \underline{u})$. This term is given by

$$\dot{\underline{p}} = \underline{v}, \tag{2}$$

$$\dot{\underline{v}} = \frac{1}{m}\left(\underline{R}_{\mathcal{W},\mathcal{B}} \begin{bmatrix} 0 \\ 0 \\ u_T \end{bmatrix} - u_T \underline{K}_{\text{drag}} \underline{v} + \underline{F}_{\text{ext}} \right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \tag{3}$$

$$\dot{\phi} = \frac{1}{\tau_\phi}(k_\phi u_\phi - \phi), \tag{4}$$

$$\dot{\theta} = \frac{1}{\tau_\theta}(k_\theta u_\theta - \theta), \tag{5}$$

$$\dot{\psi} = u_{\dot{\psi}}, \tag{6}$$

where m is the vehicle mass, $\underline{R}_{\mathcal{W},\mathcal{B}}$ is the rotation matrix from body frame $\mathcal{B}$ to inertial frame $\mathcal{W}$, $\underline{K}_{\text{drag}} = \text{diag}(k_d, k_d, 0)$ is the drag coefficient matrix, $\underline{F}_{\text{ext}}$ are the external forces acting on the vehicle (such as wind gusts), g is gravitational acceleration, and ϕ, θ, ψ represent the roll, pitch, and yaw angles of the vehicle, respectively. The values τ_ϕ, τ_θ are the time constants of the roll and pitch dynamics, respectively, and k_ϕ, k_θ are the gains of the roll and pitch dynamics, respectively. The value $u_{\dot{\psi}}$ is the heading angular rate command. Note that we

Table 3. The state estimation performance (rms error) summary.

	Hovering			Step Response			Trajectory Following			
	Indoors		Outdoors	Indoors		Outdoors	Indoors		Outdoors	Unit
Pose	**0.013**	0.019	0.043	0.118	0.133	0.091	0.091	0.097	0.099	m
x	**0.008**	0.010	0.026	0.103	0.107	0.054	0.052	0.075	0.084	m
y	**0.008**	0.014	0.028	0.028	0.048	0.062	0.062	0.078	0.048	m
z	**0.007**	**0.007**	0.019	0.050	0.062	0.041	0.042	0.065	0.022	m
Roll	**0.160**	0.322	–	–	–	–	0.857	0.345	–	°
Pitch	**0.103**	0.291	–	–	–	–	0.911	0.309	–	°
Yaw	**0.813**	0.977	–	2.704	3.789	–	0.883	0.907	–	°
Duration	30–230	50–180	20–150	30–200	20–120	20–120	30–80	25–120	50–180	s
Wind	–	11–11.5	3.6–7.4	–	11–11.5	3.6–7.4	–	11–11.5	3.6–7.4	m/s

Bold indicates the lowest rms error of each row.

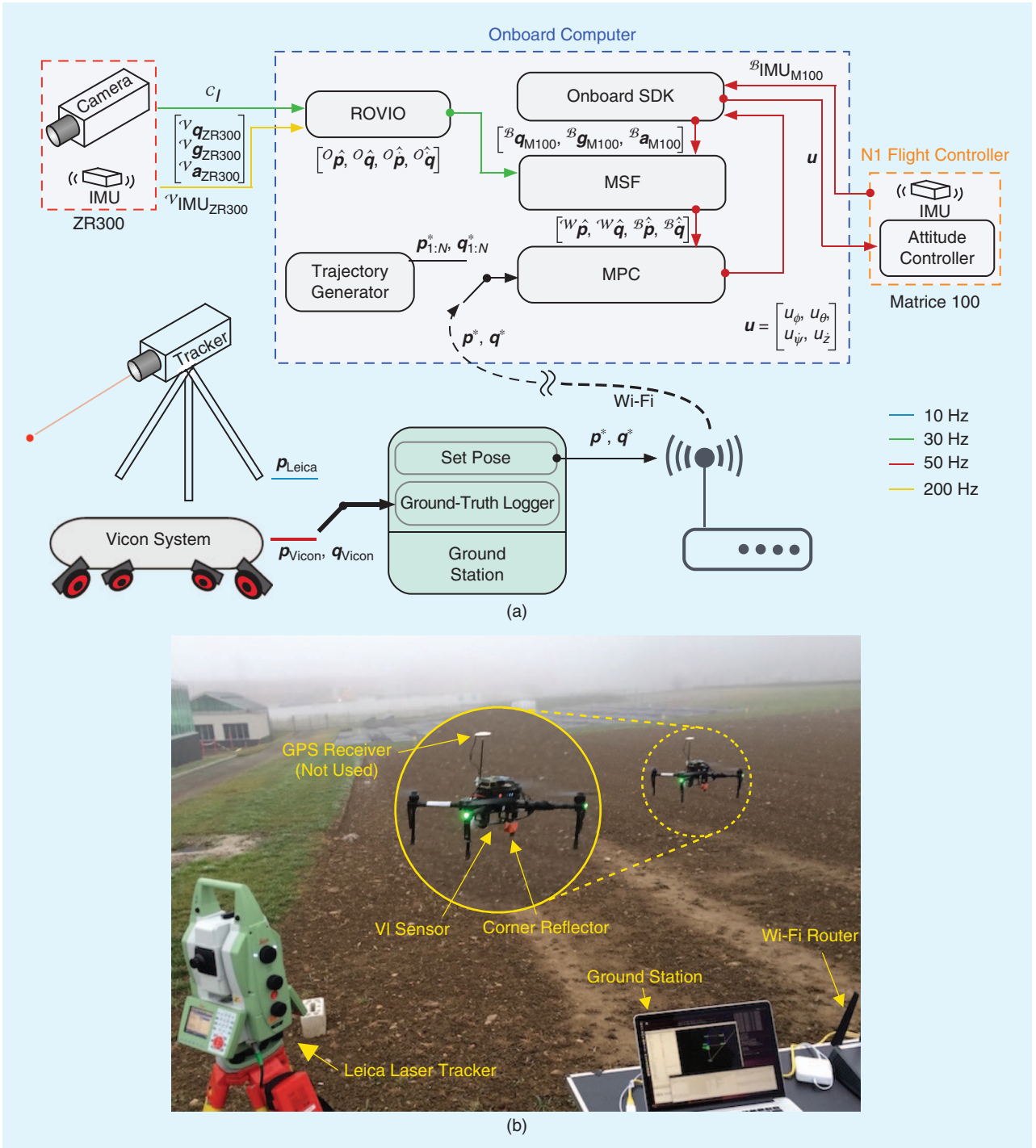

Figure 7. (a) An illustration of our software architecture, with different colors denoting the corresponding sample rates. (b) The outdoor experimental setup.

assume perfect tracking of the heading angular rate, as it does not affect the vehicle position. F_{ext} is estimated in real time using an augmented Kalman filter as described in [30].

Experimental Results

We present our implementation details and hardware and software setup and evaluate the control and state estimation performance in indoor and outdoor environments. Nine experiments are performed in different conditions while tasks are varied to demonstrate the repeatability and feasibility of the proposed approach. Tables 2 and 3 summarize the control and state estimation performances as rms errors. A detailed result analysis is presented in the following sections.

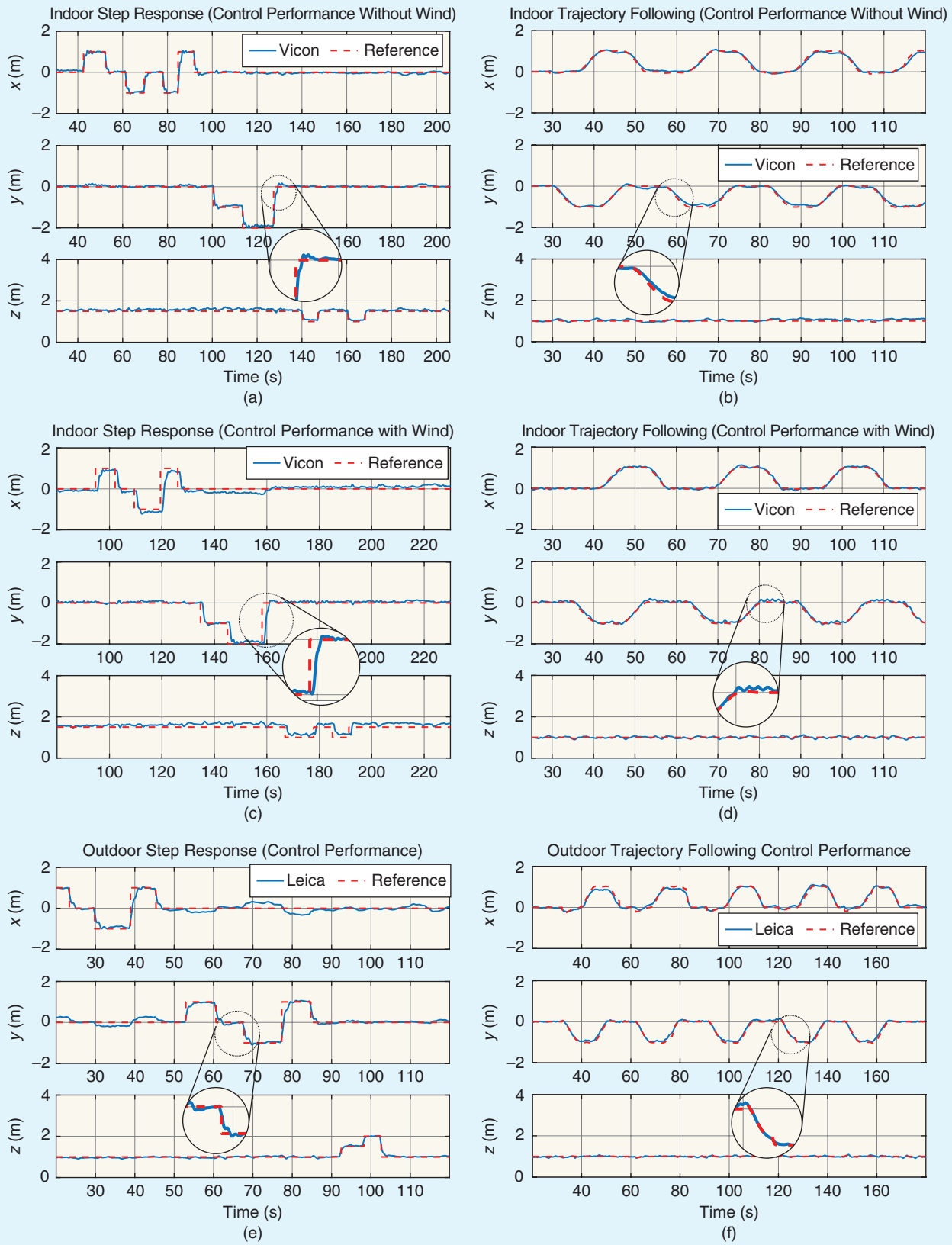

Figure 8. (a), (c), and (e) The step response and (b), (d), and (f) the trajectory-following control performance in (a)–(d) indoor and (e) and (f) outdoor environments. (g) and (h) The position and orientation control performance while the MAV tracks an aggressive trajectory seven times. *(Continued.)*

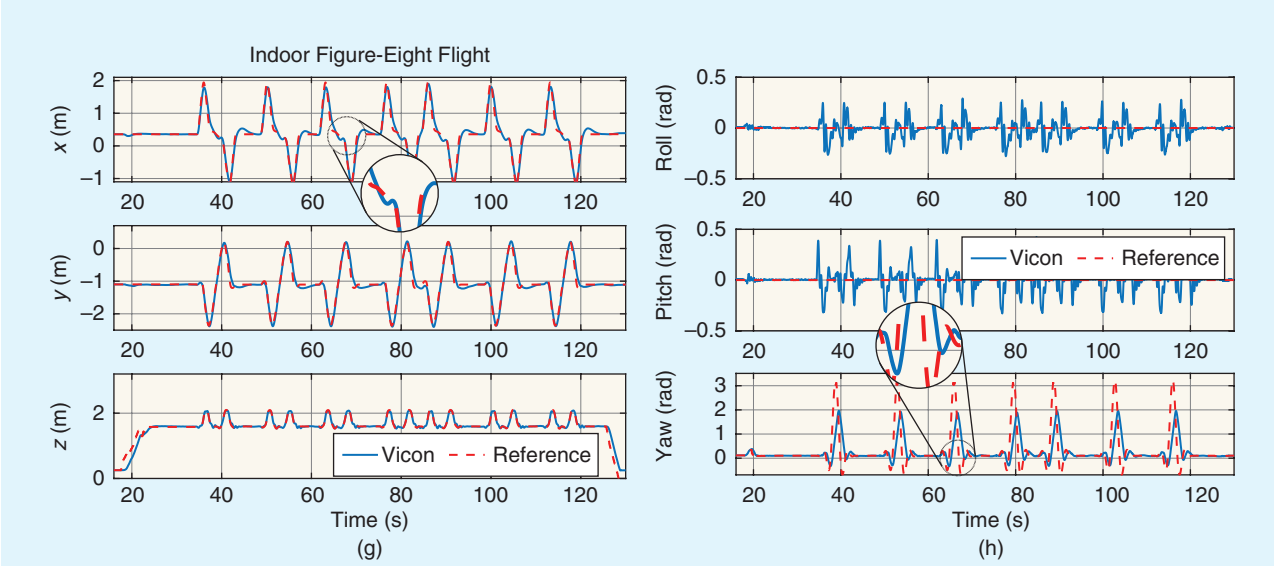

Figure 8. *(Continued.)* (a), (c), and (e) The step response and (b), (d), and (f) the trajectory-following control performance in (a)–(d) indoor and (e) and (f) outdoor environments. (g) and (h) The position and orientation control performance while the MAV tracks an aggressive trajectory seven times.

Hardware Setup

Our quadcopter MAV carries an Intel NUC 5i7RYH [i7-5557U, 3.1-GHz dual cores, 16-GB random-access memory (RAM)], running Ubuntu Linux 14.04 and ROS Indigo onboard [35]. It is equipped with a flight controller, an N1 autopilot, and an embedded IMU providing vehicle orientation, acceleration, and angular velocity at 50 Hz to the computer via 921,600 b/s universal serial bus (USB)-to-serial communication. We mount a downfacing VI sensor (the distance from C to the ground is 0.106 m sitting on a flat surface) and activate only a fisheye camera at 30 Hz and IMU at 200 Hz (disabling the stereo IR cameras and IR projector).

The total system mass is 3.62 kg, and the MAV carries 1.27-kg payload including the onboard computer, a gimbal camera, and a VI sensor. A six-cell lithium polymer battery (22.2 V, 4,500 mAh) powers the device with a total flight time of ≈12 mins without any computational load (only running the autopilot with a small angle of attack ≈±20°) and ≈10 mins, 50 s with all processes running (the low-battery threshold is set at 20% of the battery). The MAV and a ground station are connected via Wi-Fi with proper time synchronization.

Software Setup

Our system is integrated using an ROS as shown in Figure 7. Each box represents an ROS node running at different rates. ROVIO receives the fisheye images ${}^{C}\boldsymbol{I}$ and IMU measurements $[{}^{\mathcal{V}}\boldsymbol{q}_{\text{ZR300}}, {}^{\mathcal{V}}\boldsymbol{g}_{\text{ZR300}}, {}^{\mathcal{V}}\boldsymbol{a}_{\text{ZR300}}]$. The estimated odometry $[{}^{O}\hat{\boldsymbol{p}}, {}^{O}\hat{\boldsymbol{q}}, {}^{O}\hat{\dot{\boldsymbol{p}}}, {}^{O}\hat{\boldsymbol{q}},]$ is subscribed by the MSF framework [10] to increase the rate from 30 to 50 Hz using IMU from N1 flight controllers $[{}^{\mathcal{B}}\boldsymbol{q}_{\text{M100}}, {}^{\mathcal{B}}\boldsymbol{g}_{\text{M100}}, {}^{\mathcal{B}}\boldsymbol{a}_{\text{M100}}]$. The output from MSF $[{}^{\mathcal{W}}\hat{\boldsymbol{p}}, {}^{\mathcal{W}}\hat{\boldsymbol{q}}, {}^{\mathcal{B}}\hat{\dot{\boldsymbol{p}}}, {}^{\mathcal{B}}\hat{\boldsymbol{q}}]$ is fed to the subsequent MPC position controller. Finally, the control outputs $\boldsymbol{u} = [u_\phi, u_\theta, u_\psi, u_{\dot{z}}]$ are transmitted to the attitude flight controller. The ground station sets either a goal pose $[\boldsymbol{p}^*, \boldsymbol{q}^*]$ for position and orientation or N sequences $[\boldsymbol{p}^*_{1:\text{N}}, \boldsymbol{q}^*_{1:\text{N}}]$ created by the onboard trajectory generator [29].

For the indoor and outdoor experiments, we utilize the Vicon motion-capture system and the Leica laser tracker, respectively, to obtain the ground truth. Note that the Vicon system can provide the position and orientation $[\boldsymbol{p}_{\text{Vicon}}, \boldsymbol{q}_{\text{Vicon}}]$, whereas the laser tracker can only provide the position ground truth $\boldsymbol{p}_{\text{Leica}}$.

It is important to properly tune the MSF parameters, such as measurement and process noise. This particularly impacts vertical state estimation (altitude and vertical velocity) because the VI sensor is downward facing. With our setting, the MAV can fly up 15 m without experiencing issues.

Experimental Setup

We perform nine experiments in both indoor and outdoor environments, with respect to the ground truth. More specifically, three tasks are considered: hovering, step response, and trajectory following. To demonstrate controller robustness, wind disturbances are generated in the indoor environment using a fan with 260 W and 300 m^3/min air flow. As measured by an anemometer, this produces an 11–11.5 m/s disturbance in the hovering position.

To evaluate the control performance while indoors, we compute the rms error between the reference and actual vehicle positions and orientations as obtained from the motion-capture device, with the Euclidean distance used for calculating the differences between three-dimensional (3-D) poses. Outdoors, we only consider the positional error with respect to the ground truth from the laser tracker. Similarly, the state estimation performance is quantified using the rms error between the ground truth and pose estimates.

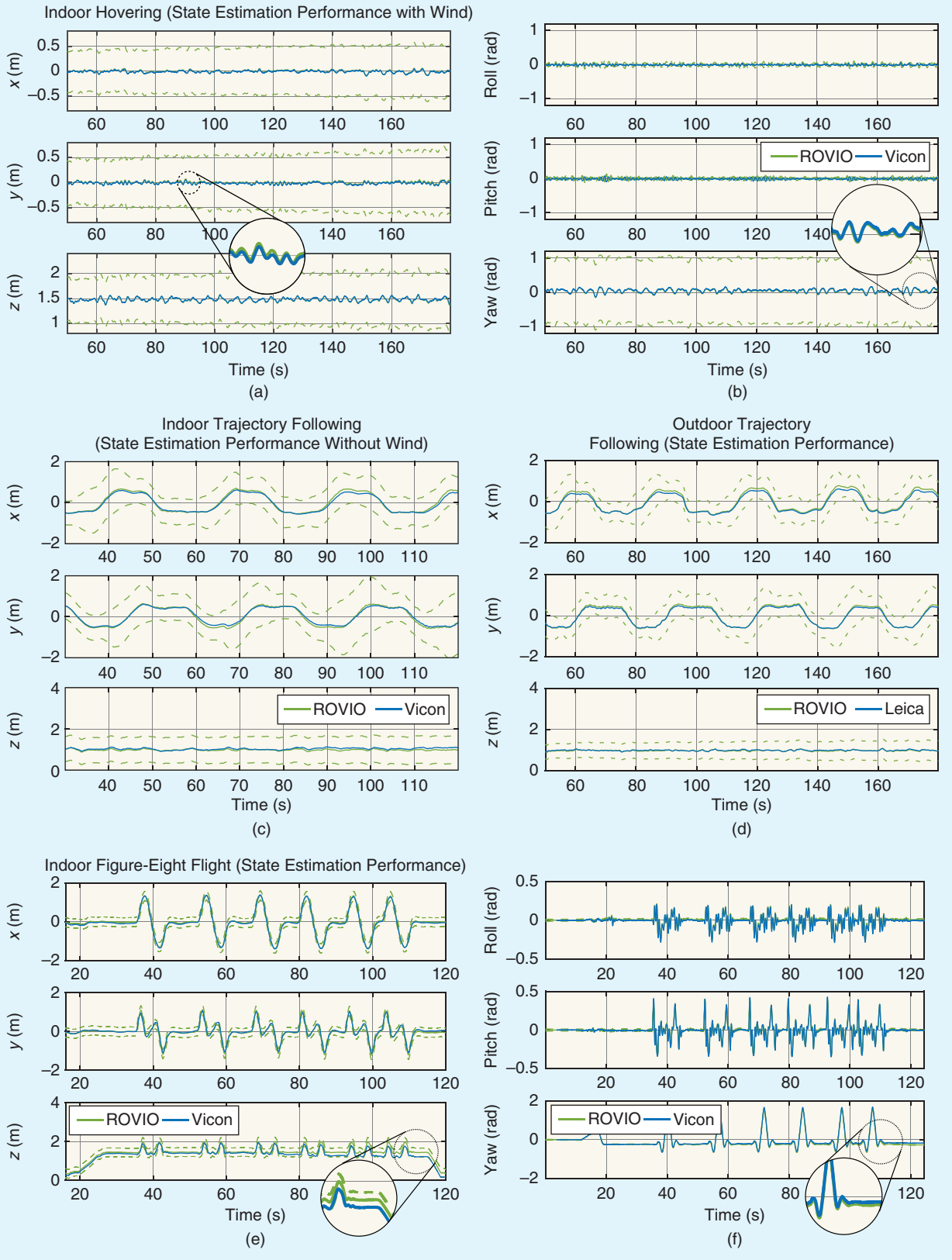

Figure 9. (a) The position and (b) orientation state estimation while hovering and the position estimation while performing the trajectory following in (c) indoor and (d) outdoor environments. (e) and (f) are position and orientation state estimations, respectively, during an aggressive trajectory following.

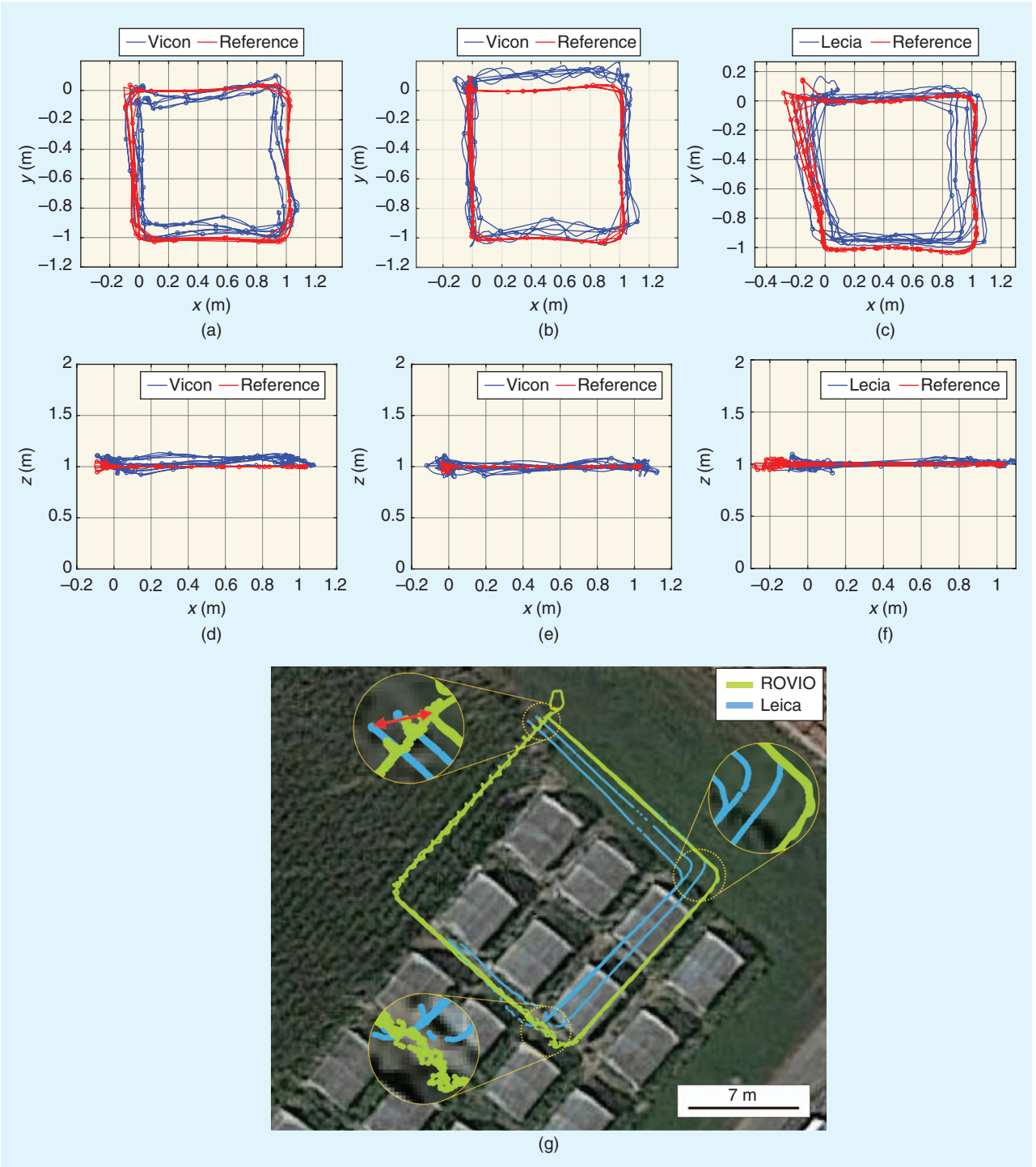

Figure 10. The qualitative results for the trajectory following in indoor and outdoor environments. A fan approximately 3 m away from the MAV was used to introduce external wind disturbances. (a) Indoors without disturbances, (b) indoors with disturbances, (c) outdoors, (d) indoors without disturbances, (e) indoors with disturbances, and (f) outdoors. (g) A longer-distance flight (≈180 m) at an outdoor farm site. Note that the laser tracker lost track midflight due to occlusions by the vehicle itself. The red arrow depicts takeoff and landing positions where a qualitative drift error is calculated. (Image courtesy of Google Maps.)

Performance Evaluation

We evaluate the control and state estimation performance evaluation using the rms error for the nine experiments. For the control and state estimation performance, the rms error between the reference commands and ground truth poses is computed for the former, and the error between the ground truth and estimated pose is used for the latter. Qualitative results are also demonstrated for short–long trajectory-following tasks. Due to space limitations, only a subset of result plots is presented. Tables 2 and 3 provide a complete result summary.

Control Performance Evaluation

Despite their resistance to external disturbances, the downside of using heavy platforms is their slower response. Figure 8 shows the step response plots without [Figure 8(a)] and with [Figure 8(c)] wind disturbances in indoor and outdoor environments [Figure 8(e)] (in the first column). A goal position is manually chosen to excite all axes. The results in Table 2 evidence a relatively large control error in both the x and y directions because slow response causes errors to accumulate as the goal reference is reached. Moreover, in Figure 9, we use the method of Burri et al. [2] to generate a smooth polynomial reference trajectory. Figure 8(b), (d), and (f) shows gentle trajectory-following control performance results, and Figure 8(g) and (h) displays a more aggressive and agile trajectory-following performances. The trajectory is configured as 10.24 m with a maximum velocity and acceleration of 1.63 m/s, and 5.37 m/s^2 given a 9.07-s time budget [36]. Note that the yaw tracking error is quite large because of physical hardware limitations (i.e., M100's maximum angular rate given the trajectory and payload). The accuracy of the trajectory following in comparison to hovering implies that the proposed approach is applicable in many practical applications, such as obstacle avoidance or path planning.

State Estimation Performance Evaluation

Figure 10(a) and (b) depicts position and orientation estimations using ROVIO and the VI sensor while hovering with wind disturbances. The plots illustrate the disturbances continuously pushing the MAV, incurring control error, whereas the estimation drifts very slowly within the $\pm 3\sigma$ boundary. Figure 10(c) and (d) shows position estimations for indoor and outdoor environments while performing the trajectory following. Note that the performance can slightly vary depending on the flying environment due to visual feature differences. Table 2 shows that the variations in the state estimations between tasks are smaller than those of the control performances. This implies that the control performance error can be caused by physical vehicle limitations (e.g., motor saturation), imperfect dynamics modeling, and manual controller tuning. The last two figure parts [Figure 10(e) and (f)] show accurate state estimations while tracking aggressive figure-eight trajectories with varying heights and yaw six times. Most states lie within the $\pm 3\sigma$ boundary, but it can be clearly seen that heights and yaw estimations drift around 110 s due to accumulated errors caused by fast maneuvers.

Qualitative Results

Figure 9 presents two qualitative results for short and long trajectory following performances. The first and second row are top and side views, respectively. Red illustrates the planned trajectory and the MAV position is marked in blue. Note that a fan is located around 3 m from the origin (i.e., $x = 0$, $y = 0$) along the southeast direction. The trajectory shift due to wind in the $+x$ and $-y$ directions is evident. The results for a long trajectory-following task are depicted in Figure 9(g). The length of one side of the square is around 15 m, and the vehicle flies around the area along the side three times (≈180 m). The plot shows that visual odometry drifts while flying at 1.288 m, with the red arrow marking the offset between take-off and landing positions. Qualitatively, the 0.82% error of the total flight distance is consistent with our previous results [16].

Conclusions

We have presented the state estimation and control performance of a VI-aided cost-effective VTOL MAV platform. The combination of robust visual odometry and a state-of-the-art controller with traditional dynamic system identification brings commercial products (MAVs and VI sensors) into the research domain. The applied methods were evaluated in both indoor and outdoor environments. Competitive results demonstrate that our approach represents a stepping stone toward achieving more sophisticated tasks. We return our experiences and findings to the community through open-source documentation and software packages to support researchers building custom VI-aided MAVs.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreements 644227 and 644128 from the Swiss State Secretariat for Education, Research, and Innovation under contract numbers 15.0029 and 15.0044.

References

[1] H. Lim, J. Park, D. Lee, and H. J. Kim, "Build your own quadrotor: Open-source projects on unmanned aerial vehicles," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 33–45, 2012.

[2] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual–inertial mapping, re-localization and planning onboard MAVs in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2015, pp. 1872–1878.

[3] S. T. Nuske, S. Choudhury, S. Jain, A. D. Chambers, L. Yoder, S. Scherer, L. J. Chamberlain, H. Cover, and S. Singh, "Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers," *J. Field Robotics*, vol. 32, no. 8, pp. 1141–1162, 2015.

[4] S. Yang, Z. Fang, S. Jain, G. Dubey, S. M. Maeta, S. Roth, S. Scherer, Y. Zhang, and S. T. Nuske, "High-precision autonomous flight in constrained shipboard environments," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-15-06, 2015.

[5] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016, pp. 5332–5339.

[6] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 971–976.

[7] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modeling, estimation and control for aerial grasping and manipulation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2011, pp. 2668–2673.

[8] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: A review," *Precision Agriculture*, vol. 13, no. 6, pp. 693–712, 2012.

[9] M. Achtelik, S. Weiss, and R. Siegwar, "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011.

[10] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments," *J. Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[11] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2096–2103, 2017.

[12] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié, and J. L. Ny. (2016). Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle. arXiv. [Online]. Available: https://arxiv.org/abs/1611.07329

[13] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual–inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 431–437.

[14] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: MIT Press, 2011.

[15] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, 2011.

[16] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2015, pp. 298–304.

[17] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. (2015). On-manifold preintegration theory for fast and accurate visual-inertial navigation. arXiv. [Online]. Available: https://arxiv.org/abs/1512.02363

[19] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual–inertial localization," in *Proc. Robotics: Science and Systems*, 2015.

[20] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual–inertial estimation with application to autonomous MAVs," in *Proc. Int. Symp. Experimental Robotics (ISER)*, 2014.

[21] R. Mur-Artal and J. D. Tardós, "Visual–inertial monocular SLAM with map reuse," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 796–803, 2017.

[22] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, Switzerland, 2007.

[23] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. New York: Springer-Verlag, 2011.

[24] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 20–32, 2012.

[25] P. Pounds and R. Mahony, "Design principles of large quadrotors for practical applications," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 3265–3270.

[26] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *Proc. AIAA Guidance, Navigation and Control Conf. and Exhibit*, 2008, p. 7410.

[27] I. Sa and P. Corke, "System identification, estimation and control for a cost effective open-source quadcopter," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 2202–2209.

[28] M. B. Tischler and R. K. Remple, *Aircraft and Rotorcraft System Identification*. Reston, VA: AIAA Education, 2006.

[29] M. Burri, J. Nikolic, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Maximum likelihood parameter identification for MAVs," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 4297–4303.

[30] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using Robot Operating System," in *Robot Operating System (ROS): The Complete Reference*, A. Koubaa, Ed. New York: Springer-Verlag, 2016, pp. 3–39.

[31] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending Kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 4304–4311.

[32] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2013, pp. 1280–1286.

[33] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[34] I. Sa, M. Kamel, R. Khanna, M. Popovic, J. Nieto, and R. Siegwart. (2017). Dynamic system identification, and control for a cost-effective and open-source multi-rotor MAV. arXiv. [Online]. Available: https://arxiv.org/abs/1701.08623

[35] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in *Proc. ICRA Workshop Open Source Software*, Kobe, Japan, 2009, vol. 3, p. 5.

[36] R. Bähnemann, M. Burri, E. Galceran, R. Siegwart, and J. Nieto, "Sampling-based motion planning for active multirotor system identification," in *Proc. 2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 3931–3938.

[37] Intel. (2017). Intel speech enabling developer kit. *Intel Corporation*. [Online]. Available: http://click.intel.com/intelr-realsensetm-development-kit-featuring-the-zr300.html

[38] Intel. (2016, Jan.). *Intel RealSense 3D Camera ZR300*. Intel Corp. [Online]. Available: http://goo.gl/DL93Wo

[39] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. (2015). *ROVIO: Robust Visual Inertial Odometry Using a Direct EKF-Based Approach*. IROS. [Online]. Available: https://goo.gl/q5zCRV

Inkyu Sa, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: inkyu.sa@mavt.ethz.ch.

Mina Kamel, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: mina.kamel@mavt.ethz.ch.

Michael Burri, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: burri210@gmail.com.

Michael Bloesch, Dyson Research Fellow, Imperial College London. E-mail: bloesch.michael@gmail.com.

Raghav Khanna, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: raghav.khanna@mavt.ethz.ch.

Marija Popović, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: marija.popovic@mavt.ethz.ch.

Juan Nieto, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: jnieto@ethz.ch.

Roland Siegwart, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland. E-mail: rsiegwart@ethz.ch.