Build your own visual-inertial odometry aided cost-effective and open-source autonomous drone

Article *in* IEEE Robotics & Automation Magazine · August 2017

DOI: 10.1109/MRA.2017.2771326

8 authors, including:

Inkyu Sa
ETH Zurich
73 PUBLICATIONS **3,598** CITATIONS
SEE PROFILE

Mina Samir Kamel
ETH Zurich
46 PUBLICATIONS **3,225** CITATIONS
SEE PROFILE

Michael Burri
ETH Zurich
30 PUBLICATIONS **5,224** CITATIONS
SEE PROFILE

Michael Bloesch
Imperial College London
69 PUBLICATIONS **6,959** CITATIONS
SEE PROFILE

Build your own visual-inertial odometry aided cost-effective and open-source autonomous drone

Inkyu Sa*, Mina Kamel, Michael Burri, Michael Bloesch,

Raghav Khanna, Marija Popović, Juan Nieto, and Roland Siegwart

Autonomous Systems Lab., ETH Zurich
Zürich, 8092, Switzerland
*inkyu.sa@mavt.ethz.ch

Abstract

This paper describes an approach to building a cost-effective and research grade visual-inertial odometry aided vertical taking-off and landing (VTOL) platform. We utilize an off-the-shelf visual-inertial sensor, an onboard computer, and a quadrotor platform that are factory-calibrated and mass-produced, thereby sharing similar hardware and sensor specifications (e.g., mass, dimensions, intrinsic and extrinsic of camera-IMU systems, and signal-to-noise ratio). We then perform system calibration and identification enabling the use of our visual-inertial odometry, multi-sensor fusion, and model predictive control frameworks with the off-the-shelf products. This approach partially circumvents the tedious parameter tuning procedures required to build a full system. The complete system is evaluated extensively 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 $0.036\,\mathrm{m}$ with respect to reference hover trajectories. We also conduct relatively long distance ($\approx$$180\,\mathrm{m}$) experiments on a farm site, demonstrating 0.82% drift error of the total flight distance. This paper conveys the insights we acquired about the platform and sensor module and offers open-source code with tutorial documentation to the community.

1 INTRODUCTION

Over the past decade, vertical taking-off and landing (VTOL) micro aerial vehicles (MAVs), which use counter-rotating 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 commercial platforms, whose perfor-

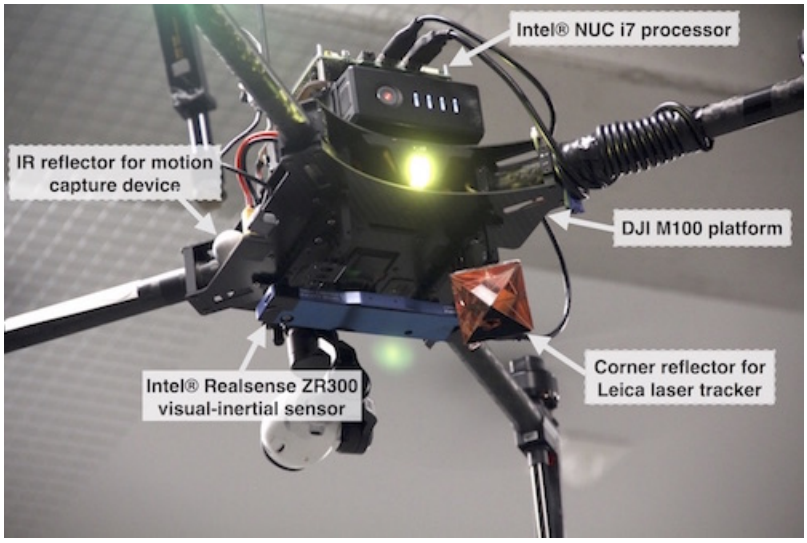


Figure 1: Matrice 100 VTOL MAV quadrotor platform with a downward-facing VI sensor. The IR reflectors and prism are mounted for obtaining ground truth.

mance has improved steadily 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, 3, 4, 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 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.

For VTOL MAVs to become more widespread, they require lower costs and more easily obtainable parts. Recently, the DJI Matrice 100 MAV[1] (Fig. 1) has been introduced as a commercial platform with parts available at local retailers. Developers can access sensor data, e.g., from the inertial measurement unit (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 such as attitude dynamics, the underlying autopilot controller structure, and input command scaling. This information is critical for subsequent position control strategies such as model predictive control (MPC).

A visual-inertial (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 time-synchronized wide field of view (FoV) image ($\approx 133^\circ$) and motion measurements. There is an impressive

[1] https://www.dji.com/matrice100

Table 1: Summary of total system cost

ID	Name	Price (USD)
1	M100[3]	1979.4
2	Visual-inertial sensor	289
3	Intel NUC i7	482
4	1TB SSD	280
5	16 GB RAM	103.5
6	TTL to USB converter (FTDI)	14.75
7	Power regulator (12V)	21
Sum		**3169.65**

research-grade visual-inertial 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[2] 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 PC.

In this paper, 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 set-ups with different inertial moments and sensor mount configurations. The contributions of this system paper are:

- a delivery of software packages (including a modified SDK, nonlinear MPC, calibration parameters, and system identification tools) with accompanying documentation to the community,
- an evaluation of control and state estimation performance in different environments,
- 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 paper. For example, the MPC strategy is applicable to other commercial VTOL platforms (e.g., Matrice 200 or 600) given identified dynamic system models using our procedures. The Robust Visual-Inertial Odometry (ROVIO) framework can also be utilized in mobile platforms for navigation tasks [14] with the VI sensor (Intel ZR300).

[2] http://click.intel.com/realsense.html

[3] You can get 40% discount with developer registration that we have to create to activate DJI's Onboard SDK (Please have a look the above link for the registration). Otherwise, we can't send commands to the N1 autopilot. We thus list 40% discounted price in the part list.

The remainder of this paper is structured as follows. Section 2 introduces the state-of-the-art in MAV VI odometry, dynamic system identification, and control. Sections 3 and 4 detail the vehicle specification, and VI odometry framework used in this work. Our system identification and control strategies are described in Section 5. We present experimental results in Section 6 before concluding in section 7.

2 Related Work

VTOL MAVs are used increasingly for tasks such as building inspection, 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 the state-of-the-art VI odometry techniques, dynamic system identification, and VTOL MAV control as most relevant sub-topics.

VI odometry has been an active research topic in the last decade given advances in microelectromechanical systems (MEMS), IMU, and imaging technologies [15]. It is a favorable option for payload-constrained platforms such as VTOL MAVs due to its light weight and relatively low computational requirements. All sensor states are jointly-coupled (i.e., tightly-coupled) within (i) filtering or (ii) 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 suffering from drift and exhibiting 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 which can be input to feedback control strategies at a constant computational cost. In contrast, the second class of methods performs keyframe-based nonlinear optimization over all jointly-coupled sensor states [17, 18, 19, 20, 21]. In some cases, these approaches can also perform loop closures to compensate for drift and provide globally consistent maps. However, they often demand costly computations that may burden smaller platforms, and demonstrate state estimation without feedback control (or only for hovering tasks).

Identifying the dynamics of attitude controllers is vital to achieving good control performance. For a common quadrotor, the rigid vehicle dynamics are well-known [22, 23] and can be modeled as a non-linear 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 non-trivial 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 CAD 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 datasets; 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 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.

3 VTOL MAV platform and visual-inertial sensor module

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

3.1 VTOL MAV platform

Our vehicle is a quadrotor with 650 mm diagonal length, and four 13 in diameter propellers with 4.5 in thread pitch. The maximum takeoff weight is 3600 g and flight time varies depending on the hardware configuration (16$\sim$40 min). The N1 autopilot manages attitude control and telemetry, but information regarding the device is not publicly disclosed. Using the SDK, sensor data can be accessed through serial communication, and we configure the IMU update rate at 50 Hz. The SDK enables access to most functionalities and supports cross-platform development environments such as the Robot Operating System (ROS), Android, and iOS. However, there is a fundamental issue in sending control commands with this protocol. The manufacturer uses `ROS services` to send commands; this is strongly not recommended[4] as they are blocking calls that should be used only for triggering signals or quick calculations. If data transaction (hand-shaking) fails (e.g., due to poor WiFi signal), it blocks all subsequent calls. Given that low control command latency ($\approx 20\,\mathrm{ms}$) can have large performance impacts, we modify the SDK to send direct control commands through the serial port. It is worth mentioning that we faced communication problems (921600 bps) 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 auto-trim 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 since 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 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, they are set as the maximum/minimum dead zone values.

[4] http://wiki.ros.org/ROS/Patterns/Communication

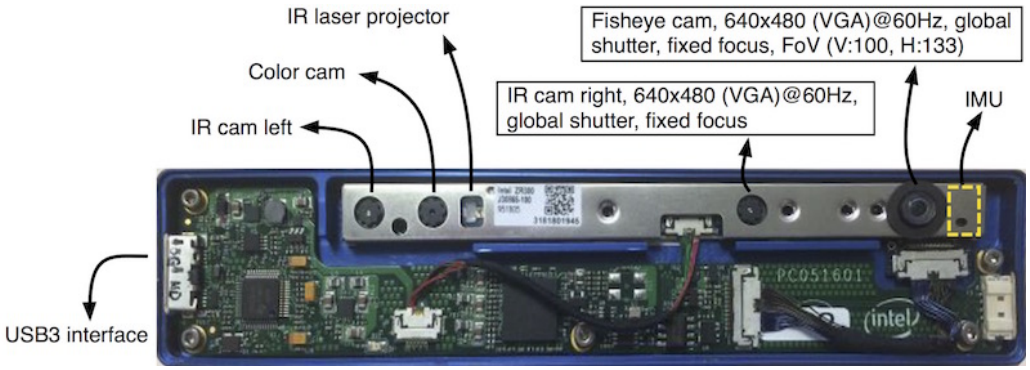


Figure 2: Intel ZR300 VI sensor module.

3.2 Visual-inertial sensor

In this paper, we exploit a ready-to-market VI sensor[5]. Most importantly, the sensor has one fisheye camera with a FoV of 133° and 100° in the horizontal and vertical directions, respectively, and streams a 640×480 image at 60 frames per second shown in Fig. 2. An onboard IMU provides 3-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, the projector, and the RGB camera, they are disabled for reduced-power operation.

Camera-IMU time synchronization for any VI related tasks is non-trivial since camera images and motions measured by the IMU are tightly connected. In the following section, we introduce the time synchronization and camera-IMU extrinsic calibration.

3.2.1 Camera-IMU time-synchronization

The ZR300 has different clock sources for the IMU and image timestamps. Therefore, direct usage of the timestamps from the RealSense library leads to poor estimator performance. To mitigate this issue, a synchronization message is generated every time the sensor captures an image, which contains the timestamp and sequence number of the corresponding image. 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 timestamp of the image with respect to the IMU before publishing it over ROS. This procedure is depicted in Fig. 3.

Another important aspect is that the sensor IMU has different sampling rates on its gyroscopes ($\sim$200 Hz) and accelerometers ($\sim$250 Hz). Since the noise density of the gyroscopes is smaller than those 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 version of the sensor, the IMU is not intrinsically calibrated. We use the extended version of

[5] http://click.intel.com/intelr-realsensetm-development-kit-featuring-the-zr300.html

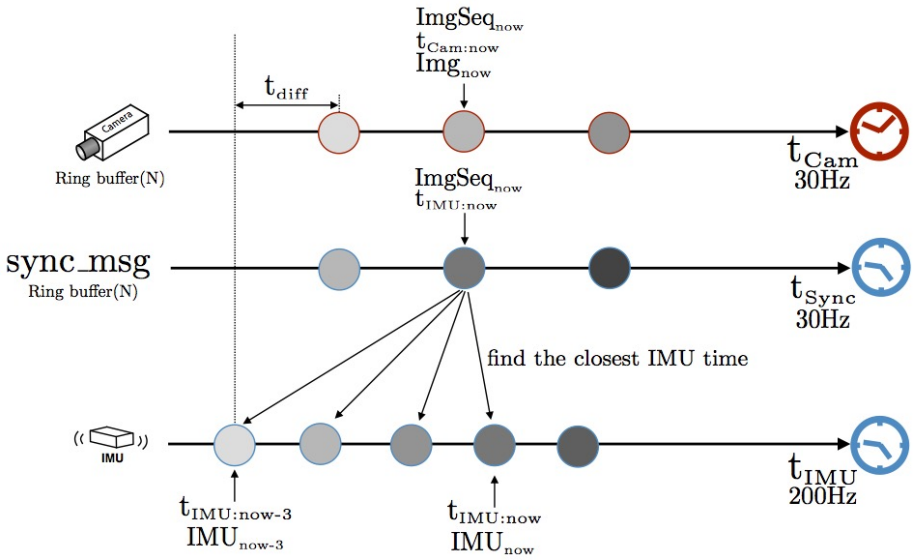


Figure 3: Camera-IMU time-synchronization illustration. Two sensors are running at different rates with own time sources as indicated by the red and blue clocks. The faster update rate of the IMU, $t_{IMU:now}$ is used as the master time. For each node, the shade of gray represents amount of time elapsed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

Kalibr [31] to estimate each axis of the gyro and the accelerometer with respect to a reference.

Lastly, we currently do not compensate for the camera exposure time. The timestamps are triggered at the beginning of the exposure time rather than in the middle. This could be important in cases of large lighting changes. Instead, we use a constant offset between the IMU and camera as estimated using Kalibr [32].

3.3 Coordinate systems definition

We define 5 right-handed frames following standard ROS convention: world $\{\mathcal{W}\}$, odometry $\{\mathcal{O}\}$, body $\{\mathcal{B}\}$, camera $\{\mathcal{C}\}$, and VI sensor IMU $\{\mathcal{V}\}$ as shown in Fig 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 Root Mean Square (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 coordinate where VI odometry is initialized. We treat them identically in this paper, but they can differ if external pose measurements (e.g., GPS or Leica laser tracker) are employed.

These coordinate systems and notations are used in the rest of this paper.

[6]Image source from http://goo.gl/7NsbmG

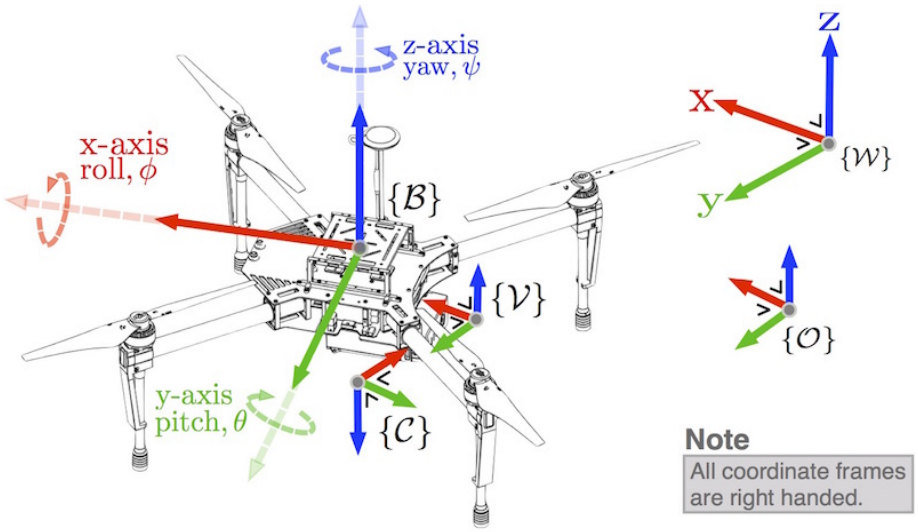


Figure 4: Coordinate system definitions[6]. There are 5 frames: world $\mathcal{W}$, odometry $\mathcal{O}$, body $\mathcal{B}$, camera $\mathcal{C}$, and VI sensor IMU $\mathcal{V}$.

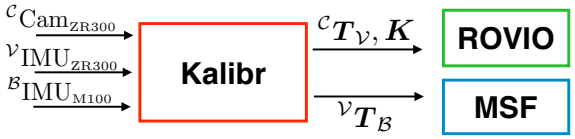


Figure 5: Three inputs of image and IMUs are fed into the Kalibr calibration framework. Intrinsic ($\boldsymbol{K}$) and extrinsic (${}^{\mathcal{C}}\boldsymbol{T}_{\mathcal{V}}$ and ${}^{\mathcal{V}}\boldsymbol{T}_{\mathcal{B}}$) are utilized by subsequent VI odometry and sensor fusion frameworks.

3.3.1 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 Fig. 5. ${}^{\mathcal{C}}\mathrm{Cam}_{\mathrm{ZR300}}$, ${}^{\mathcal{V}}\mathrm{IMU}_{\mathrm{ZR300}}$, and ${}^{\mathcal{B}}\mathrm{IMU}_{\mathrm{M100}}$ denote an image in $\{\mathcal{C}\}$ frame taken by VI sensor, IMU message in $\{\mathcal{V}\}$ from 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.

4 Visual-inertial odometry framework

This section introduces our ROVIO framework [16]. The highlights of this approach are three-fold:

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 Fig. 6), therefore computationally expensive the visual feature descriptor extraction step can be avoided.

2. It makes use of full robocentric formulation to avoid possible corruption of unobservable states. Therefore the consistency of the estimates can be improved. The landmark locations are parametrized by bearing vector and distance parameters with respect to the current camera pose in order 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 2D manifold S^2 and minimal differences/derivatives are employed for improved consistency and efficiency (the filter can be run on a single standard CPU core).

3. The IMU biases and camera-IMU extrinsics are also included into the filter state and co-estimated online for higher accuracy.

Fig. 6 illustrates an instance of feature tracking and pose estimation with ROVIO. First, a large number of key point candidates are extracted using a Fast 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 amount of features (`25` in this paper) tracked within an EKF based on local (only last a couple of frames) and global (the latest score since it has been detected last time) feature scores.

It is worth noting that ROVIO requires proper parameter tuning for optimized performance (e.g., the number of features to track, initial covariances, and inverse-depth). Although these parameters must fine-tuned for specific environments, we provide pre-tuned 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 detail, we refer the reader to our previous work on the ROVIO framework [16] for further details, including the filter setup, process and measurement models, and their update procedures.

5 Dynamic systems identification and Non-linear Model Predictive Control

In this section, we summary our recent work [34] of MAV dynamic systems identification and describe nonlinear Model Predictive Control (nMPC).

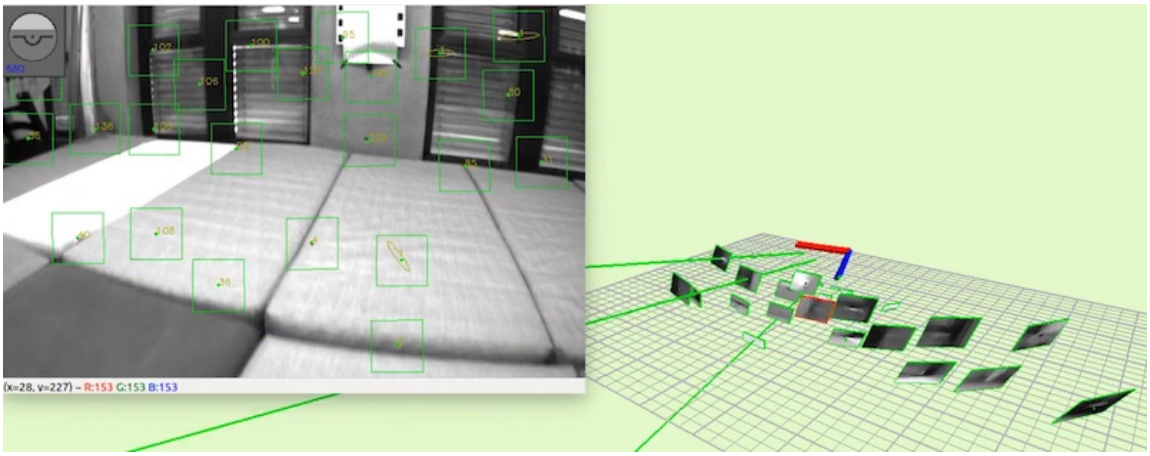


Figure 6: A snapshot of ROVIO running with a MAV dataset [33]. The left figure shows tracked landmarks (green dots on the center of green squares and their corresponding IDs). The squares are projected patches which are used for building up multi-level patches from an image pyramid. The predicted landmark location uncertainties are represented by yellow ellipses. The right figure shows 3D estimated camera poses and tracked patches including their distance uncertainties.

5.1 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 ($u_{\dot{\psi}}$ in $\mathrm{rad/s}$), 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 unit-less transmitter input commands (e.g., -1024$\sim$1024 for pitch command) to the MAV attitude response. This can be determined by linearly mapping with the maximum/minimum angles ($\pm 30^\circ$) 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].

5.2 Non-linear MPC 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 the 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 $\underline{x} = \left(\underline{p}^T, \underline{v}^T, \phi, \theta, \psi\right)^T$ where $\underline{p}$ and $\underline{v}$ are the MAV position and velocity, respectively, expressed in the 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:

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

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. $\underline{f}(\underline{x}, \underline{u})$ is the ordinary differential equation representing the MAV dynamics. This term is given by:

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

$$\dot{\underline{v}} = \frac{1}{m} \left(\underline{R}_{\mathcal{W},\mathcal{B}} \begin{bmatrix} 0 \\ 0 \\ u_T \end{bmatrix} - u_T \underline{K}_{\text{drag}} \underline{v} + \underline{F}_{\text{ext}} \right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \tag{2b}$$

$$\dot{\phi} = \frac{1}{\tau_\phi} \left(k_\phi u_\phi - \phi \right), \tag{2c}$$

$$\dot{\theta} = \frac{1}{\tau_\theta} \left(k_\theta u_\theta - \theta \right), \tag{2d}$$

$$\dot{\psi} = u_\psi, \tag{2e}$$

where m is the vehicle mass, $\underline{R}_{\mathcal{W},\mathcal{B}}$ is the rotation matrix from body frame $\mathcal{B}$ to inertial frame $\mathcal{W}$, $\underline{K}_{\text{drag}} = diag(k_{\text{d}}, k_{\text{d}}, 0)$ 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. τ_ϕ, τ_θ are the time constants of the roll and pitch dynamics, respectively. k_ϕ, k_θ are the gains of the roll and pitch dynamics, respectively. u_ψ is the heading angular rate command. Note that we assume perfect tracking of the heading angular rate as it does not affect the vehicle position. $\underline{F}_{\text{ext}}$ is estimated in real-time using an augmented Kalman Filter as described in [30].

6 Experimental results

We present our implementation details, hardware and software setup, and evaluate control and state estimation performance in indoor and outdoor environments. We perform 9 experiments in different conditions while varying tasks to demonstrate the repeatability and feasibility of the proposed approach. Tables 2 and 3 summarize the control and state estimation performances as root-mean-square (RMS) error. A detailed result analysis is presented in the following sections.

Table 2: Control performance (RMS error) summary

	Hovering			**Step response**			**Trj. following**			Unit
	Indoor		Outdoor	Indoor		Outdoor	Indoor		Outdoor	
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	—	deg
pitch	**0.793**	0.913	—	—	—	—	0.871	1.067	—	deg
yaw	1.573	3.024	—	3.659	6.865	—	**1.344**	2.858	—	deg
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

Table 3: State estimation performance (RMS error) summary

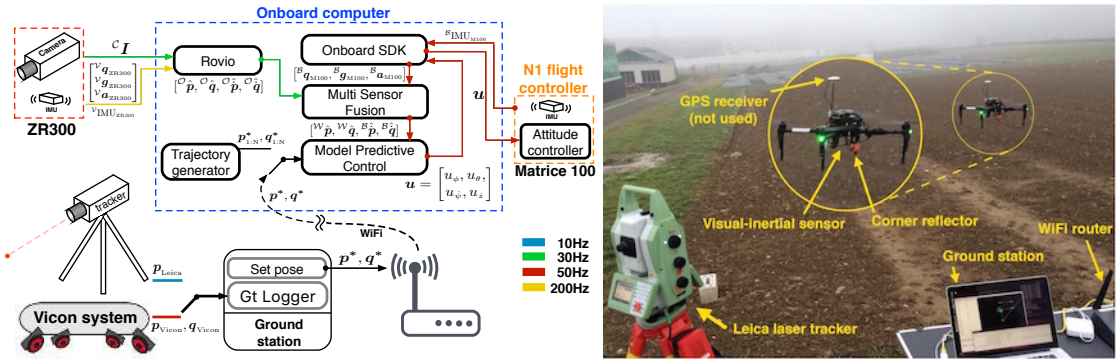
	Hovering			**Step response**			**Trj. following**			Unit
	Indoor		Outdoor	Indoor		Outdoor	Indoor		Outdoor	
Pose	**0.013**	0.019	0.043	0.118	0.133	0.091	0.091	0.097	0.099	m
x	**0.008**	0.010	0.026	0.103	0.107	0.054	0.052	0.075	0.084	m
y	**0.008**	0.014	0.028	0.028	0.048	0.062	0.062	0.078	0.048	m
z	**0.007**	**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	—	deg
pitch	**0.103**	0.291	—	—	—	—	0.911	0.309	—	deg
yaw	**0.813**	0.977	—	2.704	3.789	—	0.883	0.907	—	deg
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

6.1 Hardware Setup

Our quadcopter MAV carries an Intel NUC 5i7RYH (i7-5557U, 3.1 GHz dual cores, 16 GB RAM), running Ubuntu Linux 14.04 and ROS Indigo onboard [35]. It is equipped with a flight controller, N1 autopilot, an embedded IMU providing vehicle orientation, acceleration, and angular velocity at 50 Hz to the computer via 921,600 bps USB-to-serial communication. We mount a down-facing VI sensor (the distance from $\mathcal{C}$ 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 6-cell LiPo battery (22.2 V, 4500 mAh) powers the device with total flight time of $\approx$ 12 mins without any computational load (only running the autopilot with a small angle of attack $\approx \pm 20^{\circ}$) and $\approx$ 10 mins 50 s with all process running[7]. The MAV and a ground station are connected via WiFi with proper time synchronization.

[7] Low-battery threshold is set %20 of the battery.



(a) (b)

Figure 7: (a) illustrates our software architecture, with different colors denoting the corresponding sample rates. (b) shows our outdoor experimental setup.

6.2 Software setup

Our system is integrated using ROS as shown in Fig. 7. Each box represents a ROS node running at different rates. Rovio receives the fisheye images ${}^{\mathcal{C}}\boldsymbol{I}$ and IMU measurements $\left[{}^{\mathcal{V}}\boldsymbol{q}_{\text{ZR300}}, {}^{\mathcal{V}}\boldsymbol{g}_{\text{ZR300}}, {}^{\mathcal{V}}\boldsymbol{a}_{\text{ZR300}}\right]$. The estimated odometry $\left[{}^{\mathcal{O}}\hat{\boldsymbol{p}}, {}^{\mathcal{O}}\hat{\boldsymbol{q}}, {}^{\mathcal{O}}\hat{\boldsymbol{p}}, {}^{\mathcal{O}}\hat{\boldsymbol{q}}\right]$ is subscribed by the MSF framework [10] to increase the rate from 30 Hz to 50 Hz using IMU from N1 flight controller $\left[{}^{\mathcal{B}}\boldsymbol{q}_{\text{M100}}, {}^{\mathcal{B}}\boldsymbol{g}_{\text{M100}}, {}^{\mathcal{B}}\boldsymbol{a}_{\text{M100}}\right]$. The output from MSF $\left[{}^{\mathcal{W}}\hat{\boldsymbol{p}}, {}^{\mathcal{W}}\hat{\boldsymbol{q}}, {}^{\mathcal{B}}\hat{\boldsymbol{p}}, {}^{\mathcal{B}}\hat{\boldsymbol{q}}\right]$ is fed to the subsequent MPC position controller. Finally, the control output $\boldsymbol{u} = \left[u_\phi, u_\theta, u_{\dot{\psi}}, u_{\dot{z}}\right]$ 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 ground truth. Note that the Vicon system can provide position and orientation $[\boldsymbol{p}_{\text{Vicon}}, \boldsymbol{q}_{\text{Vicon}}]$ whereas the laser tracker can only provide 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) since the VI sensor is downwards-facing. With our setting, the MAV can fly up 15 m without experiencing issues.

6.3 Experimental setup

We perform 9 experiments in both indoor and outdoor environments to evaluate control and state estimation with respect to ground truth. More specifically, 3 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 a 11-11.5 m/s disturbance in the hovering position.

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

6.4 Performance evaluation

We evaluate control and state estimation performance evaluation using RMS error for 9 experiments. For control and state estimation performance, RMS error between 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 while Tables 2 and 3 provide a complete result summary.

6.4.1 Control performance evaluation

Despite their resistance to external disturbances, the downside of using heavy platforms is their slower response. Fig. 8 shows step response plots (a) without and (c) with wind disturbances in indoor and (e) outdoor environments (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 Fig. 10, we use the method of Burri et al. [2] to generate a smooth polynomial reference trajectory.Fig. 8 (b), (d), and (f) show gentle trajectory following control performance results, and (g) and (h) display more aggressive and agile trajectory following performance. The trajectory is configured as $10.24\,\mathrm{m}$ with maximum velocity and acceleration of $1.63\,\mathrm{m/s}$, and $5.37\,\mathrm{m/s^2}$ given $9.07\,\mathrm{s}$ time budget [36]. Note that 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 trajectory following in comparison to hovering implies that the proposed approach is applicable in many practical applications, such as obstacle avoidance or path planning.

6.4.2 State estimation performance evaluation

Fig. 9 (a) and (b) depict position and orientation estimation using Rovio and the VI sensor while hovering with wind disturbances. The plots illustrate the disturbances continuously pushing the MAV, incurring control error, whereas estimation drifts very slowly within the $\pm 3\sigma$ boundary. Fig. 9 (c) and (d) show position estimation indoors and outdoors while performing trajectory following. Note that the performance can vary slightly depending on the flying environment due to visual feature differences. Table 2 shows that variations in state estimation between tasks are smaller than those of control performance. This implies 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 figures (e) and (f) show accurate state estimation while tracking aggressive figure of 8 trajectories with varying height and yaw 6 times. Almost states lie within the $\pm 3\sigma$ boundary but it can be clearly seen that height and yaw estimation drift around $110\,\mathrm{s}$ due to accumulated errors caused by fast maneuvers.

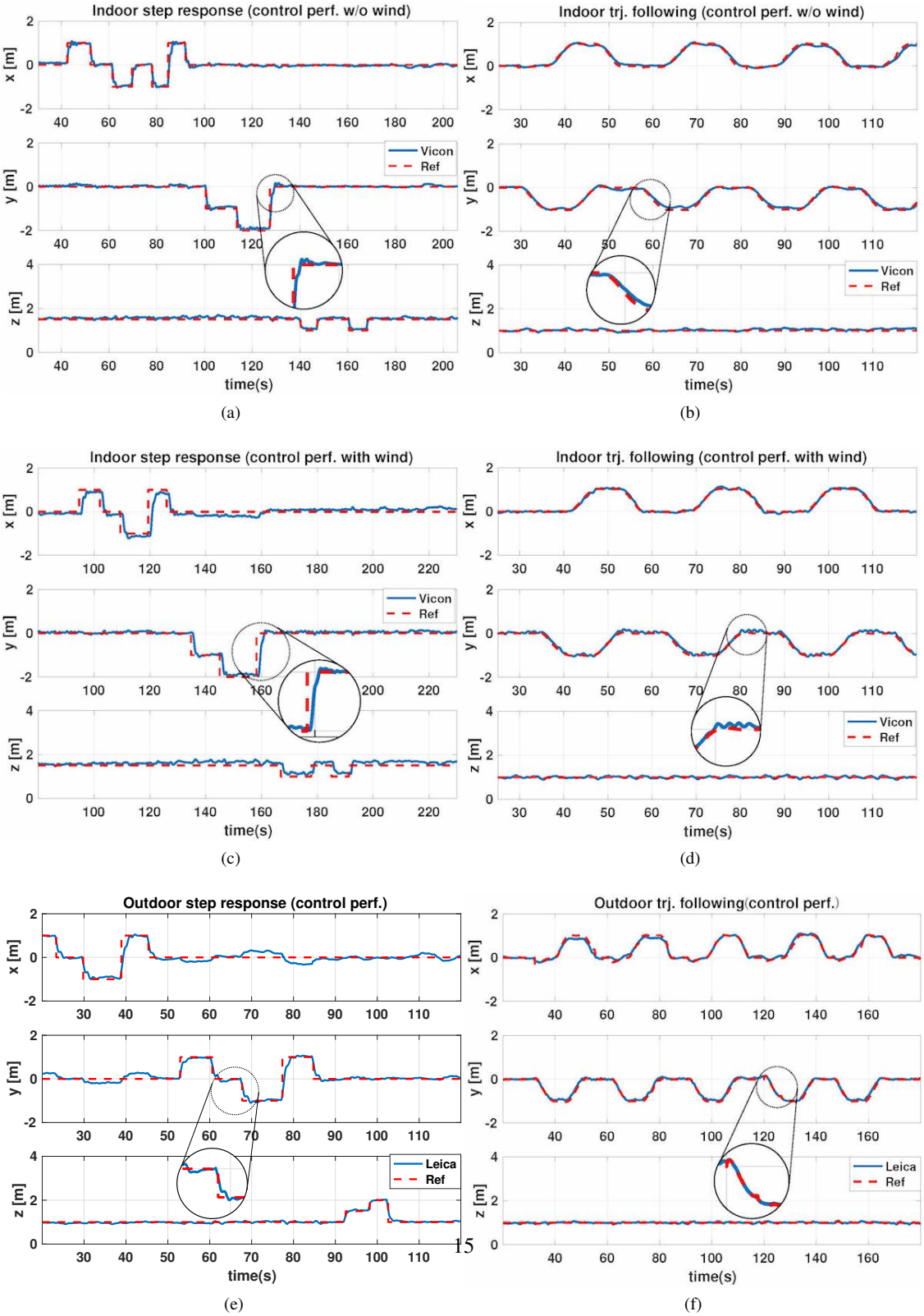
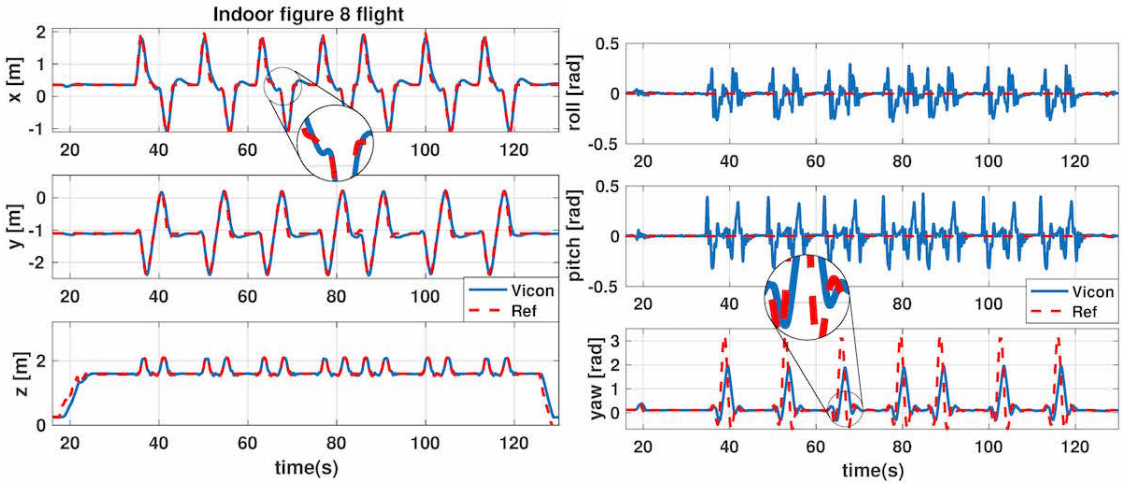


Figure 8: Step response (left column) and trajectory following (right column) control performance in indoor (a)∼(d) and outdoor environments, (e) and (f).



(g) (h)

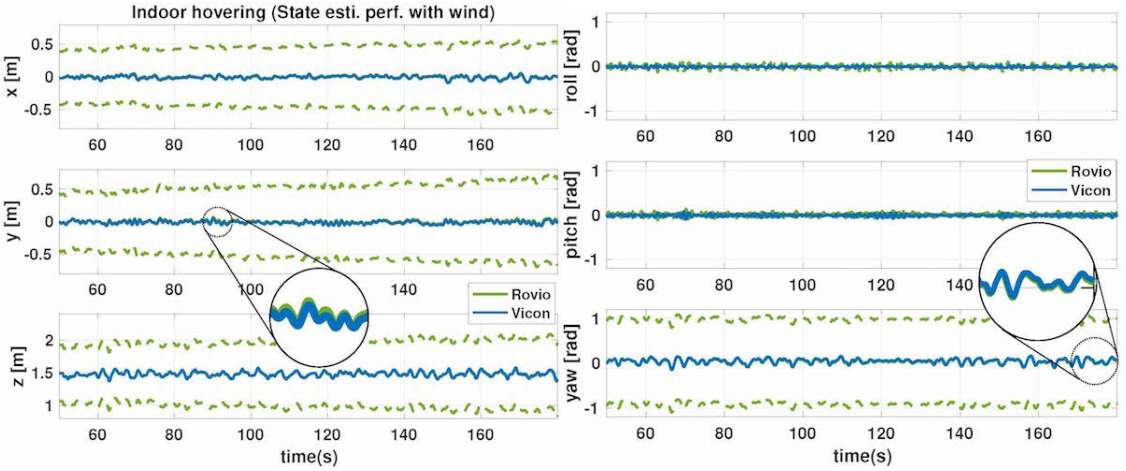
Figure 8 (Cont.): (g) and (h) show position and orientation control performance while the MAV tracks an aggressive trajectory 7 times.

6.4.3 Qualitative results

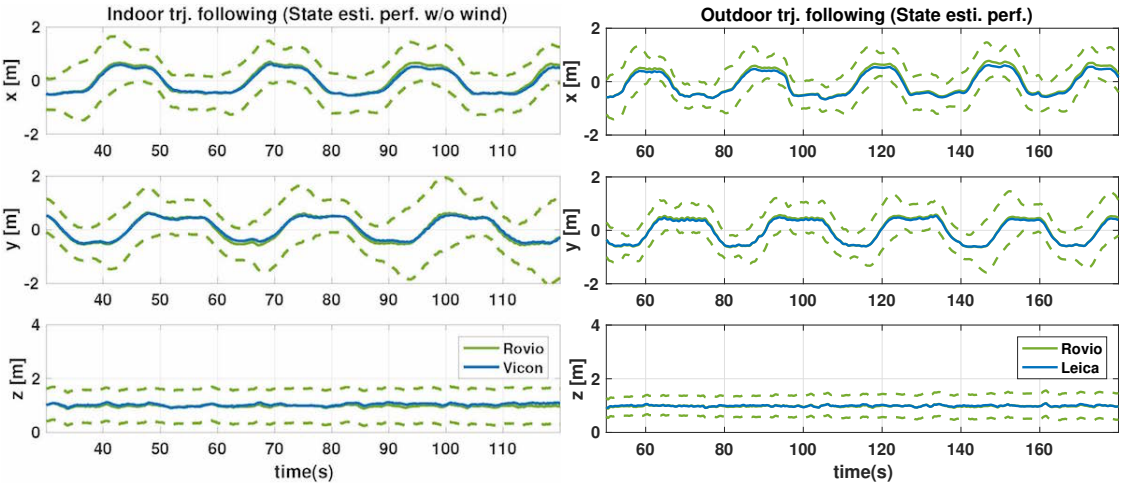
Fig. 10 presents two qualitative results for short and long trajectory following. The first and second row are top and side views, respectively. Red illustrates the planned trajectory and the MAV position is marked in blue. Note that a fan is located around 3 m away from the origin (i.e., x = 0, y = 0) along the South-East direction. The trajectory shift due to wind in the positive x- and -y-directions is evident. The results for a long trajectory following task are depicted in Fig. 10 (g). The length of one side of the square is around 15 m and the vehicle flies around the area along the side 3 times (≈ 180 m). The plot shows that visual odometry drifts while flying at 1.288 m, with the red arrow marking the offset between taking-off and landing positions. Qualitatively, the 0.82% error of the total flight distance is consistent with our previous results [16].

7 Conclusions

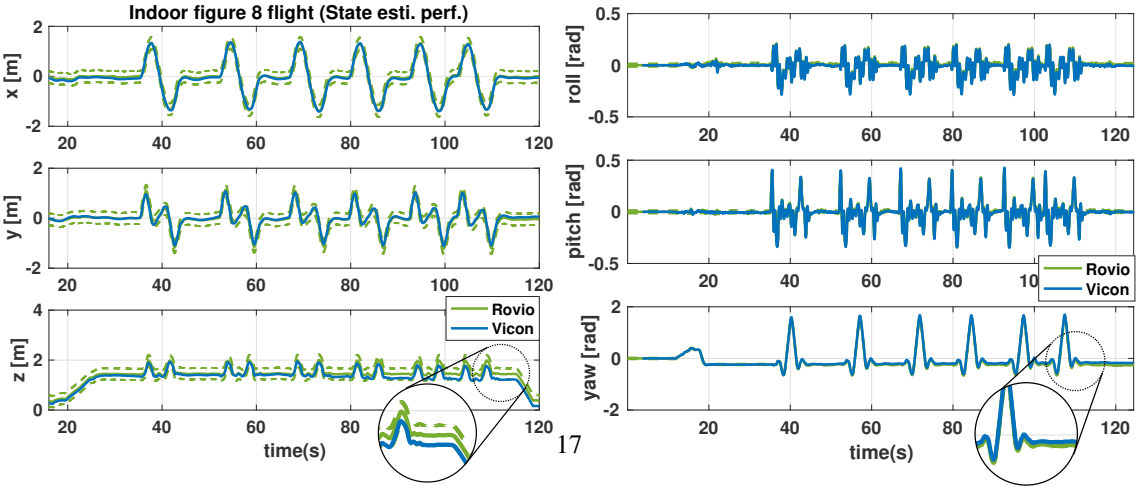
We have presented state estimation and control performance of a VI-aided cost-effective VTOL MAV platform. The combination of robust visual odometry and a state-of-the-art controller with traditional dynamic system identification brings commercial products (MAVs and visual-inertial sensors) into the research domain. The applied methods were evaluated in both indoor and outdoor environments. Competitive results demonstrate that our approach represents a stepping stone towards 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.



(a) (b)

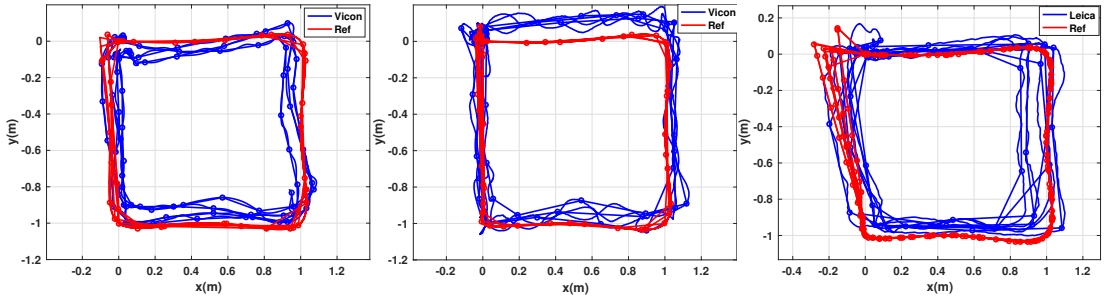


(c) (d)



(e) (f)

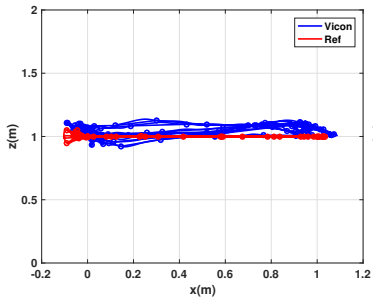
Figure 9: (a) position and (b) orientation state estimation in hover, and position estimation while performing trajectory following (c) indoors and (d) outdoors. (e) and (f) are position and orientation state estimation respectively during aggressive trajectory following.



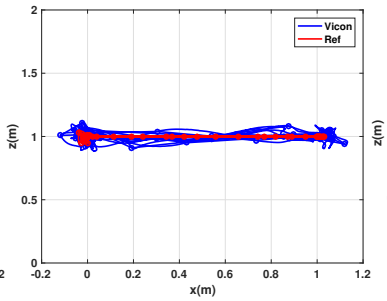
(a) Indoor w/o disturbances.

(b) Indoor with disturbances.

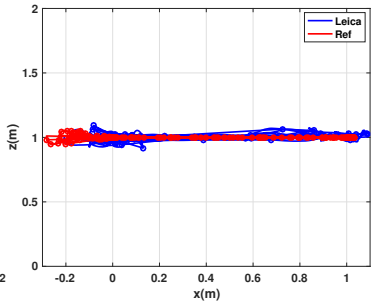
(c) Outdoor



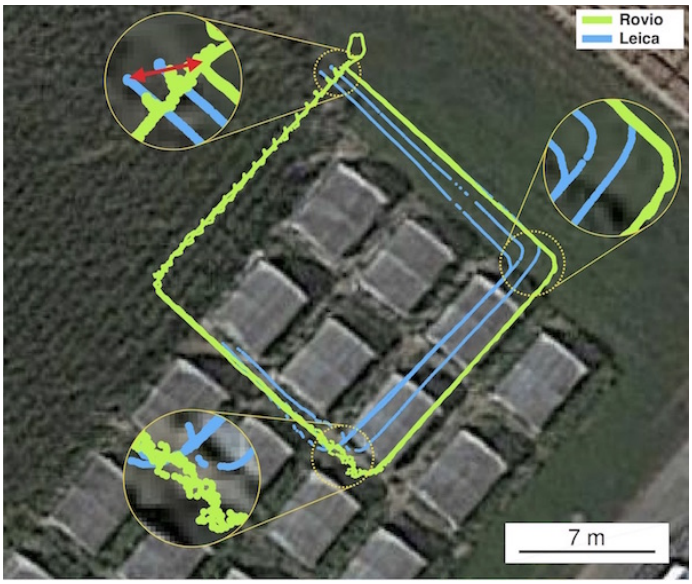
(d) Indoor w/o disturbances.



(e) Indoor with disturbances.



(f) Outdoor



(g)

Figure 10: Qualitative results for trajectory following indoors and outdoors. A fan around $3\,\mathrm{m}$ away from the MAV was used to introduce external wind disturbances. The figure in (g) shows a longer distance flight ($\approx 180\,\mathrm{m}$) in outdoor farm site. Note that the laser tracker lost track mid-flight due to occlusions by the vehicle itself. The red arrow depicts take-off and landing positions where a qualitative drift error is calculated.

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644227 and No 644128 from the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 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 Robotics and Automation Magazine*, vol. 19, no. 3, pp. 33–45, 2012.

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

[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," *Journal of Field Robotics*, June 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," Tech. Rep. CMU-RI-TR-15-06, Robotics Institute, Pittsburgh, PA, February 2015.

[5] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[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 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 971–976, IEEE, 2012.

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

[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 *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[10] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments," *Journal of 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 Robotics and Automation Letters*, 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, "Autonomous Landing of a Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle," *arXiv preprint arXiv:1611.07329*, 2016.

[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 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 431–437, IEEE, 2014.

[14] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[15] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics and Automation Magazine*, 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 298–304, IEEE, 2015.

[17] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[18] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration theory for fast and accurate visual-inertial navigation," *arXiv preprint arXiv: 1512.02363*, 2015.

[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 *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 *International Symposium on Experimental Robotics (ISER)*, 2014.

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

[22] S. Bouabdallah, *Design and control of quadrotors with application to autonomous flying*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2007.

[23] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB*, vol. 73. Springer, 2011.

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

[25] P. Pounds and R. Mahony, "Design principles of large quadrotors for practical applications," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[26] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, p. 7410, 2008.

[27] I. Sa and P. Corke, "System Identification, Estimation and Control for a Cost Effective Open-Source Quadcopter," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[28] M. B. Tischler and R. K. Remple, "Aircraft and rotorcraft system identification," *AIAA education series*, 2006.

[29] M. Burri, J. Nikolic, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Maximum likelihood parameter identification for MAVs," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016.

[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.), Springer Press, (to appear) 2016.

[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 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4304–4311, IEEE, 2016.

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

[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," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[34] I. Sa, M. Kamel, R. Khanna, M. Popovic, J. Nieto, and R. Siegwart, "Dynamic System Identification, and Control for a cost-effective and open-source Multi-rotor MAV," in *Field of Service Robotics*, 2017.

[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 *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

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