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 a 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 implies that we can partially avoid tedious parameter tuning procedures for building 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 between a reference and actual trajectories of 0.036 m, while performing hover. We also conduct relatively long distance flight (∼180 m) experiments on a farm site and achieve 0.82% drift error of the total distance flight. This paper conveys the insights we acquired about the platform and sensor module and returns to the community as open-source code with tutorial documentation.

1 INTRODUCTION

VTOL MAV platforms are aerial vehicles using counter-rotating rotors to generate thrust and rotational forces. In the past decade, these devices have acquired renown in both research and industry. Emerging applications, including structural inspection, aerial photography, cinematography, and environmental
surveillance, have spurred a wide variety of ready-to-fly commercial platforms, whose performance has experienced steady improvements in terms of flight time, payloads, and safety-related smart-features. These features allow pilots to execute different tasks in more stable, easier and safer manners. However, a key challenge is directly adapting commercial platforms [1] for robotic tasks requiring accurate dynamic models, precise, low-latency state estimators, and high-performance controllers, such as obstacle avoidance and path planning [2, 3, 4], landing on moving platforms [5], object picking [6], and precision agriculture [7].

These issues can be addressed using research-grade VTOL MAVs. For instance, Ascending Technologies provides excellent platforms [8, 9] dedicated to advanced aerial applications. There is also a well-explained software development kit (SDK), sufficient scientific resources such as self-contained documentation, and online support. However, their relatively high cost may pose issues for researchers, and replacing parts after crashes (which can occur in early development stages) is difficult due to the limited number of retailers.

For VTOL MAVs to become more pervasive, they require lower costs and more easily replaceable parts. Recently, the DJI Matrice 100 MAV\(^1\) (Fig. 1) has been introduced as a commercial platform with parts available from 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 [10]. Although the manufacturer provides documentation, there is still a lack of essential scientific resources concerning 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 sensor is a favorable choice for aerial robotics; mainly due to its light-weight, low-power consumption, and ability to recover unknown scales (monocular camera). This sensor suite can provide time-

\(^1\)https://www.dji.com/matrice100

Figure 1: Matrice 100 VTOL MAV quadrotor platform with a downward-facing visual inertial sensor. The IR reflectors and the prism are mounted for obtaining ground truth.
synchronized wide field of view (FoV) image (≈ 133°) and motion measurements. There is an impressive research-grade visual-inertial sensor [11] providing time-synchronized and calibrated IMU-stereo camera images with supporting technical documentation and device drivers. Unfortunately, this sensor is relatively expensive and its production was discontinued. The Intel ZR300\footnote{http://click.intel.com/realsense.html} has recently emerged as an interesting visual-inertial sensor alternative. It is an affordable and mass-produced module that permits applying the same configuration and calibration parameters to other sensors expecting low-performance penalties. The total cost for visual-inertial sensor, drone, and PC is summarized in Table 1.

In this paper, we address these gaps by using a ready-to-market affordable VTOL platform and a visual-inertial sensor to perform system identification, sensor calibration, and system integration. The system can autonomously track trajectories or poses with high precision in indoor and outdoor environments utilizing onboard sensors. This fundamental ability enables extensions to any high-level applications, allowing researchers to efficiently build systems without tedious parameter tuning procedures. Moreover, we provide self-contained documentation and tutorials 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 modified SDK, nonlinear MPC, calibration parameters, and system identification tools) and their documentation to the community,
- An evaluation of control and state estimation performance both indoors and outdoors with ground truth,
- A demonstration of use-cases adapting our approach to build own research platforms and presenting a compete step-by-step tutorial 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) as long as their systems

---

\[\text{Table 1: Summary of total cost}\]

| ID | Name                                | Price (USD) |
|----|-------------------------------------|-------------|
| 1  | M100\footnote{You can get 40% discount with developer registration that we have to create in order to activate DJI's Onboard SDK. Otherwise, we can’t send commands to the N1 autopilot. We thus list 40% discounted price in the part list.} | 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     |
are identified (i.e., dynamics and input commands scaling) following our procedures. Rovio can also be utilized in mobile platforms for navigation tasks with the visual-inertial sensor (Intel ZR300).

The remainder of this paper is structured as follows. Section 2 introduces the state-of-the-art in MAV visual-inertial odometry, dynamic systems identification and control. Section 3 and 4 describes the vehicle specification, and visual-inertial odometry framework used in this paper respectively. Our system identification and control strategies are presented in following section 5. We present our experimental results in section 6 before concluding in section 7.

2 Related Work

In both industry and research, VTOL MAVs are gaining popularity for tasks such as building inspection, aerial photography, and precision agriculture. To perform these missions, functionalities such as 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 visual-inertial odometry techniques, dynamic system identification, and VTOL MAV control as most relevant sub-topics.

Visual-inertial odometry has been an active research topic in the last decade given advances in microelectromechanical systems (MEMS), IMU, and imaging technologies [12]. It is a favorable option for payload-constrained aerial platforms such as VTOL MAVs mainly 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 the ego-motion of the robot by integrating the visual and inertial measurements while keeping the preprocessing of the sensor data to a minimum and thereby improving the stochastic consistency. Filter based approaches often do this within a Kalman Filter (e.g. [13]), thus suffering from drift and exhibiting only a limited representation of the global environment. However, the achieved localization accuracy suffices for stabilizing control and the execution of local tasks. Furthermore, these frameworks procure constant computational cost for onboard computation and estimates which can be fed into subsequent feedback control strategies. In contrast, the second class of methods performs keyframe-based nonlinear optimization over all jointly-coupled sensor states [14, 15, 16, 17, 18]. In some cases, these approaches can also perform loop closures to compensate for drift and provide globally consistent maps. However, they often demand expensive computations that may burden smaller platforms, and only demonstrate state estimation without feedback control (or only for hovering tasks).

Identifying the dynamic systems and behaviors underlying attitude controllers is vital to achieving good control performance. For a common quadrotor, the rigid vehicle dynamics are well-known [19] and can be modeled as a non-linear system with individual rotors attached to a rigid airframe, accounting for drag force and blade flapping [20]. 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 [21, 22]. A linear least-squares method is used to estimate parameters from recorded flight data in batch offline processing [23, 24]. This approach only requires flight datasets; however, identification must be repeated if the vehicle configuration changes. Online system identification involves applying recursive estimation to real-time flight data. Burri et al. [25] demonstrate 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 aerodynamics parameters such as rotor thrust, rotor moment, and drag coefficients. We follow a batch-based approach to determine the dynamic vehicle parameters from short manual pilot maneuvers. This allows us to obtain the parameters required for MPC [26] 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 the sensor module (Realsense 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 (16~40 min). The N1 autopilot manages attitude control and telemetry, but information regarding the device is not publicly disclosed. Various sensor data can be accessed using the SDK 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 as it is a blocking call that should be used only for triggering signals or quick calculations. If data transaction (hand-shaking) fails for some reason (e.g., poor WiFi connection), it blocks all subsequent calls. Given that small control command latency ≈ 20 ms have large performance impacts, we modify the SDK to send direct control commands via serial communication. It is worth to mention that we faced an issue of serial communication (921600 bps) between N1 autopilot and onboard computer while receiving/sending data at 100 Hz. The communication is getting unstable (i.e., we cannot send any command to N1) at that frequency. We are investigating this issue but all experiments presented in this paper use 50 Hz.

There are usually trim switches from an ordinary transmitter that allow for small command input adjustments. The N1 autopilot, however, has auto-trim functionality that balances attitude by estimating horizontal velocity. This feature 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 aspect of the autopilot is the presence of a dead zone in the small range close to the neutral value where the autopilot ignores all input commands. This function is also useful for manual piloting since the vehicle should not react to small inputs from hand tremors, but it significantly degrades control performance. We determine this by sweeping control commands around the dead zone and detecting the control inputs when any motion is generated (i.e., horizontal and vertical velocity changes). Although

\[4\]http://wiki.ros.org/ROS/Patterns/Communication
this task is difficult with a practical 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, we set them as the maximum/minimum dead zone values.

3.2 Visual-inertial sensor

In this paper, we exploit a ready-to-market visual-inertial sensor\(^5\). Most importantly, the sensor has one fisheye camera with a FoV of 133° and 100° horizontal and vertical 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 are disabled for reduced-power operation.

Camera-IMU time synchronization for any visual-inertial 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 provided by the RealSense library leads to poor estimator performance. To mitigate this problem, a synchronization message is generated every time the sensor captures an image. This message contains the timestamp and the sequence number of the corresponding image. The same sequence number is also contained in the image struct. 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 illustrated in Fig. 3.

Another important aspect is that the sensor IMU has different sampling rates on its gyroscopes (~200 Hz) and accelerometers (~250 Hz). Since the noise density of the gyroscopes is smaller than those of the ac-

\(^5\)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 depicted red and blue clocks. Faster update rate of IMU time, $t_{IMU:now}$ is used as the master time. The level of gray intensity of each node represents amount of time elapsed (i.e., the same intensity means the same time). $N$ denotes the size of ring buffers.

celometers 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 we linearly interpolate the accelerometer messages.

In our present version of the sensor, the IMU is not intrinsically calibrated. We use the extended version of Kalibr [27] to estimate each axis of the gyro and the accelerometer with respect to a reference.

Lastly, we currently do not compensate for the exposure time of the camera. The timestamps are triggered at the beginning of the exposure time rather than in the middle. This could be important in the presence of large lighting changes. Instead, we use a constant offset between the IMU and camera which is estimated using Kalibr [28].

3.3 Coordinate systems definition

We define 5 right-handed frames following standard ROS convention: world $\{W\}$, odometry $\{O\}$, body $\{B\}$, camera $\{C\}$, and visual-inertial sensor IMU $\{V\}$ as shown in Fig 4. $\{B\}$ 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 $\phi$, pitch $\theta$, and yaw $\psi$ about the x, y, and z-axis respectively for 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 $\pi$ along the x-axis to align with $\mathcal{B}$. 

7
There are 5 frames: world $\mathcal{W}$, odometry $\mathcal{O}$, body $B$, camera $C$, and visual-inertial sensor IMU $V$.

The $\{\mathcal{W}\}$ and $\{\mathcal{O}\}$ frames are fixed coordinate where visual-inertial odometry is initialized. We treat them identical in this paper, but they can differ if external pose measurements (e.g., GPS or Leica laser tracker) are employed.

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

### 3.3.1 Extrinsic calibration

Two essential transformations are required before a flight: 1) the transformation from the vehicle IMU frame $\{B\}$ to the visual-inertial sensor’s camera frame $\{C\}$, i.e., $B_T^C \in SE(3) \subset \mathbb{R}^{4 \times 4}$, and 2) the transformation from the camera frame $\{C\}$ to visual-inertial sensor’s IMU frame $\{V\}$, $C_T^V$. These extrinsic parameters and camera intrinsics are identified with Kalibr [28]. Rovio and MSF make use of this calibration data as shown in Fig. 5. $\mathcal{C}_{Cam_{ZR300}}$, $\mathcal{V}_{IMU_{ZR300}}$, and $\mathcal{B}_{IMU_{M100}}$ denote an image in $\{C\}$ frame taken by visual inertial sensor, IMU message in $\{V\}$ from visual inertial sensor, and IMU message in $\{B\}$ from Matrice 100’s autopilot respectively. $K$ is an intrinsic camera parameter describing lens surface using equidistant distortion model. Note that this procedure only needs once if visual inertial sensor’s configuration (e.g., change in position and orientation) is changed. Otherwise, pre-calibrated configuration can be used without complex calibration process.

---

6Image source from http://goo.gl/7NsbmG
Figure 5: Three inputs of image and IMUs are fed into Kalibr calibration framework. Intrinsic ($K$) and extrinsic ($^C T_V$, $^V T_B$) are utilized by subsequent visual odometry and sensor fusion frameworks.

4 Visual-inertial odometry framework

This section introduces our visual-inertial odometry framework Rovio [13]. The highlights of Rovio are three-fold:

(i) Our approach directly leverages pixel intensity errors as innovation term inside the Extended Kalman Filter and thereby results in a tighter fusion of inertial and visual sensor data. The framework employs multilevel image patches as landmark descriptors (see Fig. 6), therefore computationally expensive visual feature descriptor extraction step can be avoided.

(ii) Rovio makes use of full robocentric formulation to avoid possible corruption of the 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).

(iii) 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 amount of key point candidates is 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 the removal, The adapted Shi-Tomasi score that takes into account of the combined Hessian on multiple scales is computed for each candidate. The higher Shi-Tomasi score implies the better alignment of the candidate to the corresponding multilevel patch feature. Feature bucketing technique manages to ensure 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 framework based on local (only last a couple of frames) and global (the latest score since it has been detected last time) feature scores.

It worths to remark that Rovio requires proper parameter tuning for improved performance (e.g., the number of features to track, initial covariances and inverse-depth). Although these parameters have to be checked and fine-tuned based on environments where the drone is operating, we provide pre-tuned parameters for the camera-IMU (Realsense ZR300) and the MAV (Matrice 100) in office-like indoor and farm-site environments.
Due to the requirements of The IEEE Robotics & Automation Magazine, we refer the reader to find our previous work of visual-inertial odometry framework [13] for detailed formulations including filter setup, process and measurement models and their update.

5 Dynamic systems identification and Non-linear Model Predictive Control (nMPC)

In this section, we summary our recent work [30] of MAV dynamic systems identification and describe nonlinear Model Predictive Control.

5.1 Dynamic systems identification

Our nMPC controller requires first-order attitude (roll and pitch), thrust dynamics for position control and second-order dynamics for the disturbance observer. To identify these dynamic systems, we record input and output data while manual flight. The input are transmitter input commands, roll angle \( u_\phi \) in \( \text{rad} \), pitch angle \( u_\theta \) in \( \text{rad} \), yaw rate \( u_\psi \) in \( \text{rad/s} \), and thrust \( u_z \) in \text{N}. Output corresponds to vehicle’s response such that position, orientation, linear and angular velocities provided by a motion capture system. Input and output are logged on an onboard computer.

After the data collection, we perform input commands scale estimation that maps unitless transmitter input commands (e.g., -1024~1024 for pitch command) to vehicle’s attitude response. It 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., unbalanced platform and subtle dynamical difference. Therefore, we estimate these scaling parameters using nonlinear least-squares optimization.

Followed by input scale estimation, we use classic system identification techniques given input and output without time delay estimation option and estimated dynamic systems are presented in [30].

5.2 Non-linear MPC for full control

The vehicle controller is based on nonlinear MPC [26]. A cascade approach is used where a high level nonlinear MPC is generating attitude commands to a low level attitude controller running on the MAV autopilot. The dynamic behavior of the attitude controller is considered in the high level controller as a first order system. The state vector is defined as \(\bar{x} = (p^T, v^T, \phi, \theta, \psi)^T\) where \(p\) and \(v\) are respectively the MAV position and velocity expressed in the inertial frame, \(\{W\}\). We also define the control input vector as \(u = (u_\phi, u_\theta, u_T)^T\). We also define the reference state at time \(t\), \(\bar{x}_{ref}(t)\), and the steady state control input at time \(t\), \(\bar{u}_{ref}(t)\). Every time step, the following optimization problem is solved:

\[
\begin{align*}
\min_{\bar{u}} & \int_{t=0}^{T} (\bar{x}(t) - \bar{x}_{ref}(t))^T Q_{\bar{x}} (\bar{x}(t) - \bar{x}_{ref}(t)) + (\bar{u}(t) - \bar{u}_{ref}(t))^T R_{\bar{u}} (\bar{u}(t) - \bar{u}_{ref}(t)) dt \\
& + (\bar{x}(T) - \bar{x}_{ref}(T))^T P (\bar{x}(T) - \bar{x}_{ref}(T)) \\
\text{subject to} & \quad \dot{\bar{x}} = f(\bar{x}, \bar{u}); \\
& \quad \bar{u}(t) \in U_C \\
& \quad \bar{x}(0) = \bar{x}(t_0) .
\end{align*}
\]

where \(Q_{\bar{x}} \geq 0\) is the penalty on the state error, \(R_{\bar{u}} \succ 0\) is the penalty on control input error and \(P\) is the terminal state error penalty. The \(\succeq\) operator denotes positive definiteness of a matrix. \(f(\bar{x}, \bar{u})\) is the ordinary differential equations representing the dynamics of the MAV, which is given by:

\[
\begin{align*}
\dot{p} &= v, \\
\dot{v} &= \frac{1}{m} \left( R_{WB} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} - u_T K_{\text{drag}} v + F_{\text{ext}} \right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \\
\dot{\phi} &= \frac{1}{\tau_\phi} (k_\phi u_\phi - \phi), \\
\dot{\theta} &= \frac{1}{\tau_\theta} (k_\theta u_\theta - \theta), \\
\dot{\psi} &= u_\psi,
\end{align*}
\]

where \(m\) is the mass of the vehicle, \(R_{WB}\) is the rotation matrix from body frame \(B\) to inertial frame \(W\), \(K_{\text{drag}} = \text{diag}(k_d, k_d, 0)\) is the drag coefficients matrix, \(F_{\text{ext}}\) is the external forces acting on the vehicle (such as wind gusts), \(g\) is the gravity acceleration, \(\phi, \theta, \psi\) represent the roll, pitch and yaw angles of the vehicle. \(\tau_\phi, \tau_\theta\) are respectively the time constant of roll and pitch dynamics. \(k_\phi, k_\theta\) are respectively the gains of roll and pitch dynamics. \(u_\psi\) is the heading angular rate command. Note that we assume perfect tracking...
Table 2: Control performance (RMS error) summary

|                      | Hovering | Step response | Trj. following | Unit |
|----------------------|----------|---------------|----------------|------|
|                      | Indoor   | Outdoor       | Indoor         | Outdoor | Indoor | Outdoor |
| Pose                 | 0.036    | 0.049         | 0.233          | 0.395   | 0.260  | 0.083   | 0.091 | 0.100 | m |
| x                    | 0.016    | 0.022         | 0.155          | 0.277   | 0.189  | 0.066   | 0.042 | 0.079 | m |
| y                    | 0.018    | 0.012         | 0.125          | 0.230   | 0.172  | 0.039   | 0.071 | 0.056 | m |
| z                    | 0.026    | 0.038         | 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 | 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         | 0.118          | 0.133   | 0.091  | 0.097   | 0.099 | m |
| x                    | 0.008    | 0.010         | 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 | m |
| z                    | 0.007    | 0.007         | 0.019          | 0.050   | 0.062  | 0.041   | 0.042 | 0.065 | 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 | 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 |

of the heading angular rate as it does not affect the MAV position. $F_{\text{ext}}$ is estimated in real-time using an augmented Kalman Filter as described in [26].

6 Experimental results

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

The quadcopter carries an Intel NUC 5i7RYH (i7-5557U, 3.1 GHz dual cores, 16 GB RAM), running Ubuntu Linux 14.04 and ROS Indigo onboard [31]. The quadcopter is equipped with a flight controller, N1, 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 visual-inertial sensor and activate only a fisheye camera at 30 Hz and IMU at 200 Hz (Stereo IR cameras and IR projector are disabled).

The total system mass is 3.62 kg and the vehicle carries 1.27 kg payload including the onboard computer, a gimbal camera, and a visual-inertial sensor. A 6-cell LiPo battery (22.2 V, 4500 mAh) powers the vehicle and the total flight time is around 12 mins without any computation (only autopilot is running with a small angle of attack ≈ ±20°) and about 10 mins 50 s with all process running. WiFi connection is established between the vehicle and a ground station with proper time synchronization.

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 \( I \) and IMU measurements \([v q_{ZK300}, v g_{ZK300}, v a_{ZK300}]\). The estimated odometry \([\hat{O}^p, \hat{O}^q, \hat{O}^\dot{p}, \hat{O}^\dot{q}]\) is subscribed by the Multi-Sensor Fusion (MSF) framework [9] to increase the rate from 30 Hz to 50 Hz using IMU from N1 flight controller \([B q_{M100}, B g_{M100}, B a_{M100}]\). The output from MSF, \([W^p, W^q, B^\dot{p}, B^\dot{q}]\) is fed to the subsequent MPC position controller. Finally, the control output \(u = [u_\phi, u_\theta, u_\dot{\psi}, u_\dot{z}]\) are transmitted to the attitude flight controller.

The ground station sets either a goal position, denoted as \([p^*, q^*]\), for position and orientation or N sequences, \([p^*_1:N, q^*_1:N]\), created by the onboard trajectory generator [25].

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 \([p_{Vicon}, q_{Vicon}]\) whereas the laser tracker can only provide position ground truth \(p_{Leica}\).

It is important to properly tune MSF’s parameters such as measurement and process noise. This particularly impacts on vertical state estimation (altitude and vertical velocity) since the visual inertial sensor is facing down. With our setting, the MAV can fly up 15 m without an issue.

6.3 Experiments Setup

We perform 9 experiments in both indoor and outdoor environments to evaluate control and state estimation using ground truth. More specifically, 3 tasks are conducted; hovering, step response, and trajectory following. To demonstrate controller robustness, wind disturbances are generated in the indoor experiments using a fan that has 260 W and 300 m³/min air flow. As measured by an anemometer, this produces a 11-11.5 m/s disturbance at the hovering position.

\footnote{Low-battery threshold is set \( \%20 \) of the battery.}
Figure 7: (a) illustrates software packages diagram and different colors denote the corresponding sample rate. (b) shows experimental setup in outdoor tests.

For control performance evaluation, we use the root-mean-square (RMS) error metric between the reference and actual vehicle positions and orientations obtained by the motion capture device in indoor tests. Euclidean distance is used for 3D pose RMS error calculation. For outdoor experiments, the laser tracker provides only position ground truth. Similarly, state estimation performance is evaluated using RMS error between ground truth and pose estimates.

6.4 Performance evaluation

We present quantitative control and state estimation performance evaluation via 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 Table 2 and 3 summarize all experimental results.

6.4.1 Control performance evaluation

Despite offering strong resistance to external disturbances, the downside of using a heavy platform is 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 Table 2 results depict a relatively large control error in both x and y, because slow response causes accumulating error as the goal reference is reached. We use the method of Burri et al. [2] to generate a smooth polynomial reference trajectory as shown in Fig. 10. Even though hovering and step responses explicitly demonstrate control performance, trajectory following is also an important task in many applications. 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 m with maximum velocity and acceleration of 1.63 m/s, and 5.37 m/s² given 9.07 s time budget [32]. Note that yaw tracking error is quite large because of physical hardware limitations (i.e., M100’s maximum angular rate given trajectory
and payload). The accurate results of these reliable and repeatable trajectory following in comparison to hovering implies that the proposed approach is applicable for 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 visual-inertial sensor while hovering with wind disturbances. The plots illustrate that disturbances continuously push the vehicle, 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) illustrate 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 s due to accumulated errors stemmed from fast maneuvers.

### 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 vehicle 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 as indicated by the icon. 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 ($\approx 180$ 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 [13].

### 7 Conclusions

We have presented state estimation and control performance of a visual-inertial 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 through both indoor and outdoor experiments with ground truth. 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 own visual-inertial aided MAVs.
Figure 8: Step response (left column) and trajectory following (right column) control performance in indoor (a)∼(d) and outdoor environments, (e) and (f).
Figure 8 (Cont.): (g) and (h) show position and orientation control performance while the drone tracks aggressive trajectory 7 times.

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, re-localization 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.
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.
Figure 10: Qualitative results for trajectory following indoors and outdoors. A fan around 3 m away from the vehicle is indicated by the icon. (g) shows a longer distance flight ($\approx 180$ 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.
[5] 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.

[6] 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.

[7] 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.

[8] M. Achtelik, S. Weiss, and R. Siegwart, “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.

[9] 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.

[10] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié, and J. L. Ny, “Autonomous Landing of a Multicopter Micro Air Vehicle on a High Velocity Ground Vehicle,” arXiv preprint arXiv:1611.07329, 2016.

[11] 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.

[12] D. Scaramuzza and F. Fraundorfer, “Visual odometry [tutorial],” IEEE Robotics and Automation Magazine, vol. 18, no. 4, pp. 80–92, 2011.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.
[19] S. Bouabdallah, *Design and control of quadrotors with application to autonomous flying*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2007.

[20] 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.

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

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

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.