Dynamic System Identification, and Control for a cost-effective and open-source Multi-rotor MAV

Inkyu Sa, Mina Kamel, Raghav Khanna, Marija Popović, Juan Nieto, and Roland Siegwart

Abstract This paper describes dynamic system identification, and full control of a cost-effective Multi-rotor micro-aerial vehicle (MAV). The dynamics of the vehicle and autopilot controllers are identified using only a built-in IMU and utilized to design a subsequent model predictive controller (MPC). Experimental results for the control performance are evaluated using a motion capture system while performing hover, step responses, and trajectory following tasks in the presence of external wind disturbances. We achieve root-mean-square (RMS) errors between the reference and actual trajectory of x=0.021 m, y=0.016 m, z=0.029 m, roll=0.392°, pitch=0.618°, and yaw=1.087° while performing hover. Although we utilize accurate state estimation provided from a motion capture system in an indoor environment, the proposed method is one of the non-trivial prerequisites to build any field or service aerial robots. This paper also conveys the insights we have gained about the commercial vehicle and returned to the community through an open-source code, and documentation.

1 INTRODUCTION

Multi-rotor MAV platforms are rotorcraft air vehicles that use counter-rotating rotors to generate thrust and rotational forces. These vehicles have become a very popular research and commercial platform during the past decade. The wide variety of ready-to-fly platforms today is proof that they are being utilized for real-world aerial tasks such as indoor and outdoor inspection [1], aerial photography, cinematography, and environmental survey and monitoring for agricultural applications. The performance of these vehicles has also shown steady improvement over time in terms of flight time, payloads, and safety-related smart-features. However, it is challenging to adapt these commercial platforms for robotic tasks such as obstacle avoidance and path planning [2,15], object picking [3], and precision agriculture [4]. Because an accurate dynamics model, a low-latency and precise state estimator, and a high-performance controller are required to perform these tasks.

Ascending Technologies provides excellent research grade MAV platforms [5] [6] dedicated to advanced aerial robotic applications. There is also a well-explained software development kit (SDK) and abundant scientific resources. These platforms

Inkyu Sa
ETH Zurich, Autonomous Systems Lab., Zurich, e-mail: inkyu.sa@mavt.ethz.ch

Mina Kamel
ETH Zurich, Autonomous Systems Lab., Zurich e-mail: mina.kamel@mavt.ethz.ch

are ideal for developing aerial robots. However, their relative expensive cost may be a hurdle for researchers and replacing parts in a case of a crash (which can occur in the early development stage) is time-consuming due to the limited number of retail shops.

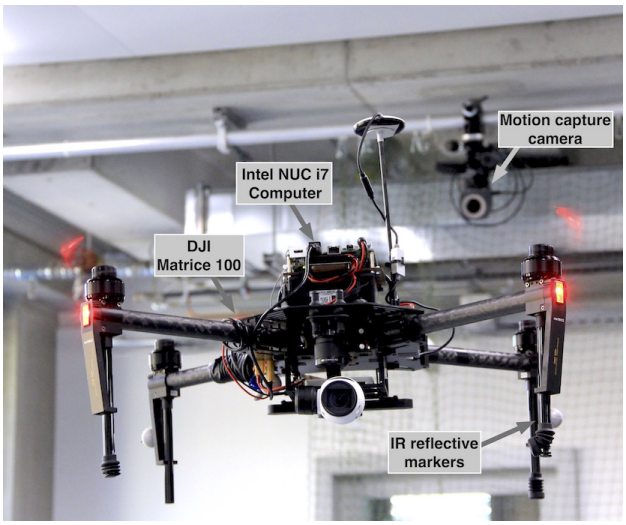


Fig. 1 A commercial Multi-rotor MAV quadrotor platform (Matrice 100). An onboard computer is mounted, and the external motion capture device provides position, velocity, and orientation measurements at 100 Hz for Model Predictive Controller (MPC).

For vertical take-off and Landing (VTOL) MAVs to become more pervasive, they must become lower in cost and their parts easier to replace. There is an affordable consumer grade VTOL platform shown in Fig. 1. Developers can now access sensor data such as IMU and barometers and send command data to the low-level attitude controller. It is easy and prompt to order parts from local retail stores with short delivery spans. There are, however, difficulties in using these platforms for robotics applications (e.g., lack of essential scientific resources such as attitude dynamics and the structure of underlying autopilot controller). This information is essential for the development of aerial robots. In this paper, we address these gaps by performing system identification using only the built-in onboard IMU. Researchers can perform their system identification with the provided documentation and build a base platform for field and service aerial robots.

The contributions of this system paper are:

- Presenting full dynamics system identification that is employed by a subsequent MPC position controller enabling to build a research grade field and service aerial robotic platform.
- Delivering software packages including modified SDK, linear MPC, and system identification tools and their documentation to the community.

`http://goo.gl/lXRnU8`

The benefit of this paper would be that the proposed techniques can be directly applied to other products such as Matrice 200, or 600 series.

The remainder of this paper is structured as follows. Section 2 introduces state-of-the-art work on MAV system identification and control and Section 3 describes the specification of the vehicle, system identification, and control strategies. We present our experimental results in Section 4, and conclude in Section 5.

2 Related Work/Background

VTOL MAVs' popularity is gaining momentum both industry and research field. It is necessary to identify their underlying dynamics system and behavior of attitude controllers to achieve good control performance. For a common quadrotor, the rigid vehicle dynamics are well-known [7] and can be modeled as a non-linear system with individual rotors attached to a rigid airframe, taking into account of drag force and blade flapping [8]. In practice, however, the identification of attitude controllers is often a non-trivial task for such consumer products due to the lack of scientific resources.

There are system identification techniques to estimate dynamic model parameters in literature. Traditionally, parameter estimation has been performed offline using complete measurement data obtained from a physical test bed and CAD models [9,10]. Such pioneer offline methods significantly contributed to the VTOL MAV community in the early development stage. Recently, Burri et al. [11] demonstrated a method for identification of the dominant dynamic parameters of a VTOL MAV using Maximum Likelihood approach. Alternatively, a linear least-squares method is used to estimate parameters from recorded flight data in batch processing manner [12, 13]. We follow a batch-based approach to determine the dynamic parameters of the vehicle from short manual pilots. This allows us to obtain the parameters necessary for MPC using only the onboard IMU and without applying any restrictive simplifying assumptions. Given the identified dynamics model, we use a high-performance state-of-the-art Model Predictive Control (MPC) [14] for horizontal position control.

3 Matrice 100 VTOL MAV platform and control

In this section, we present overviews for the hardware platform, software development toolkit, and address coordinate systems, attitude dynamics, and control strategy.

We define 2 right-handed frames following standard Robot Operating System (ROS) convention: world $\{\mathcal{W}\}$ and body $\{\mathcal{B}\}$ shown in Fig 2. The x-axis in $\{\mathcal{B}\}$ indicates forward direction for the vehicle, y-axis is left, and z-axis it up. We use Euler angles; roll (ϕ), pitch (θ), and yaw (ψ) about x, y, z-axes respectively for the RMS error calculation and visualization purposes. Quaternions are utilized for any computational processes. The defined coordinate systems and notations are used over the rest of paper.

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

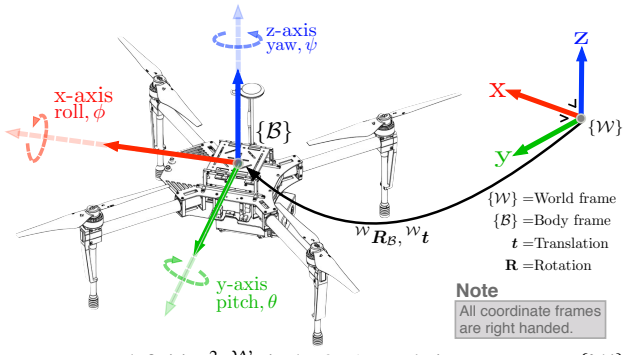


Fig. 2 Coordinate systems definition[2]. $^{\mathcal{W}}t$ is the 3×1 translation vector w.r.t $\{\mathcal{W}\}$. $^{\mathcal{W}}R_{\mathcal{B}}$ is 3×3 matrix and rotates a vector defined w.r.t $\{\mathcal{B}\}$ to a vector w.r.t $\{\mathcal{W}\}$.

3.1 Aerial platform and its SDK

The general hardware specification of Matrice 100 is well documented, this section only highlights our findings. The vehicle is a quadrotor and has 650 mm diagonal length. It uses N1 flight controller, but the information regarding the device is not disclosed to the public. The variety of sensing data can be accessed using SDK through serial communication, such as IMU, GPS, barometer, and magnetometer. The SDK enables to access most functionalities and supports cross-platform development environments. We use the onboard SDK with the Robot Operating System (ROS) wrapper, but there is a fundamental issue for sending control commands with this protocol. The manufacturer uses `ROS services` to send commands that are strongly not recommended[3]. It is a blocking call that should be used for triggering signals or quick calculations. If data transaction (hand-shaking) remains as a failure for some reasons (e.g., poor WiFi connection), it blocks all subsequent calls. Small latency $\approx$ 10 ms in control commands makes a huge difference in the resultant performance. We thus modify the SDK to send direct control commands via serial communication.

Common control commands through a transmitter are pitch, roll angles (rad), yaw rate (rad/s), and thrust (N). However, the vertical stick input of the platform is velocity (m/s) that permits easier and safer manual flight. To address this different, we use a classic PID vertical position controller alongside linear MPC horizontal position controller. Interestingly, there is no trim buttons on the provided transmitter.; instead, the N1 autopilot has auto-trim functionality (e.g., position mode) that balances attitude by estimating horizontal velocity. This feature allows easier and safer manual piloting but introduces a constant offset position error for controlling. This needs to be properly compensated. We estimate the balancing point where the vehicle's motion is minimum (hovering) and then adjust the neutral position to the estimated balancing point. If there is a change in an inertial moment (e.g., mounting a new device or changing the battery position), the balancing position has to be updated.

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

Another interesting aspect of the autopilot is the presence of relatively large dead zone that is the range close to the neutral value. Within this zone, the autopilot ignores all input commands, and no API is supported to set this. This function is also useful for a manual pilot since the vehicle should not react to small inputs yielded by tremor of hands but significantly degrades the performance of the controller. We determine this by sweeping control commands around dead zone area and detecting the control inputs when any motion is generated. Although this task is difficult with a real VTOL platform due to its fast and naturally unstable dynamics, we use the manufacture's hardware-in-loop simulator (DJI Assistant 2)[4] that enables to receive input commands from the transmitter. After determine the dead zone, we simply compensate it by adding the estimated offset to control commands if they lie within the dead zone range. Details of the modifications and tutorial are given in `http://goo.gl/vSCQjg`.

3.2 Dynamic systems identification

In this section, we present full dynamics system identification resulting from the simulator and experiments. We record input and output data; Virtual RC commands and attitude response while manual flight on an onboard computer.

3.2.1 Input commands scaling

Prior to performing system identification, it is necessary to identify the relation between Virtual RC (actual control commands) and the corresponding attitude measurements from the IMU. This can be determined by linearly mapping with the maximum/minimum angles ($\pm 30^\circ$), however there is small error in practice. This can be caused by a variety of sources such as unbalanced platform, small dynamics modelling error. We estimate these parameters using nonlinear least-squares optimization such that

$$\lambda^\star := \underset{\lambda}{argmin} \sum_{k=1}^{T} \| z_k^{\text{Meas}} - \lambda u_k^{\text{cmd}} \|^2 \tag{1}$$

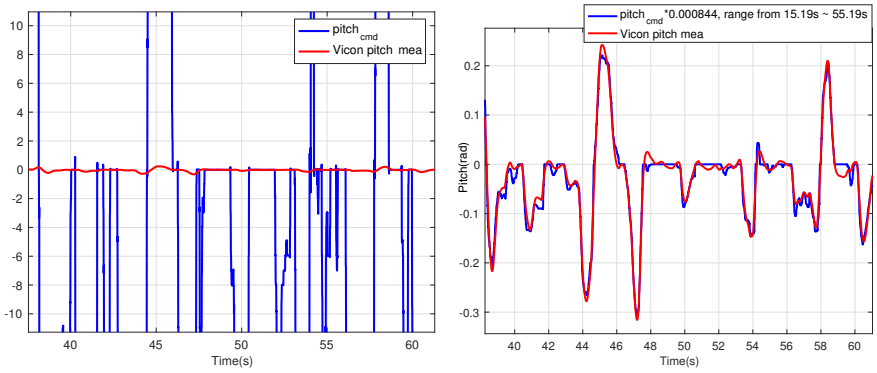
where T denotes the number of samples used for optimization. λ is 4×1 vector containing roll, pitch, and yaw rate scaling parameters, $[\lambda_\phi, \lambda_\theta, \lambda_\psi, \lambda_z]^T$. z_k^{Meas} is 4×1 of $[\phi, \theta, \psi, \dot{z}]^T$ obtained from IMU and a motion capture device at 100 Hz. u_k^{cmd} is also 4×1 vector of Virtual RC commands $[u_\phi, u_\theta, u_\psi, u_z]^T$. Note that it is difficult to measure vertical velocity using IMU, but there are three options; 1) motion capture device, 2) ultrasonic, barometer, and IMU fusion for vertical velocity estimation from the onboard flight controller, 3) using the simulator. The first approach is the most accurate but the third is useful and produces similar results as shown Table 1. The input commands and output measurements are aligned with cross-correlation to remove the delay between the two signals. This is acceptable since we estimate the signal magnitude. Fig. 3 (a) shows original input command in blue and angle

[4] http://goo.gl/Pk5kTL

measurement in red. (b) displays results after scaling using estimated parameters from Table 1.

Scale param	Experiment	Simulator
λ_ϕ	8.65×10^{-4}	8.35×10^{-4}
λ_θ	8.44×10^{-4}	8.23×10^{-4}
λ_ψ	2.24×10^{-3}	3.23×10^{-3}
$\lambda_{\dot{z}}$	2.65×10^{-3}	3.02×10^{-3}

Table 1 Virtual RC scaling parameters



(a) (b)

Fig. 3 u_ϕ scaling parameter estimation. (a) and (b) are before and after the scaling.

3.2.2 Roll and pitch attitude dynamics

Our linear MPC controller [14] requires first order attitude dynamics for position control and the second order for the disturbances observer. We estimate the dynamics by recording the input and output at 100 Hz and logged two sets of a dataset for model training and validation.

We assume a low-level flight controller that can track the reference roll, ϕ^*, and pitch, θ^*, angles with first order behavior. The first order approximation provides sufficient information to the MPC to take into account the low-level controller behavior. We thus utilize classic system identification techniques such that:

$$\frac{y(s)_\phi}{u(s)_\phi} = \frac{3.544}{s+2.118}, \quad \frac{y(s)_\theta}{u(s)_\theta} = \frac{3.827}{s+2.43}$$

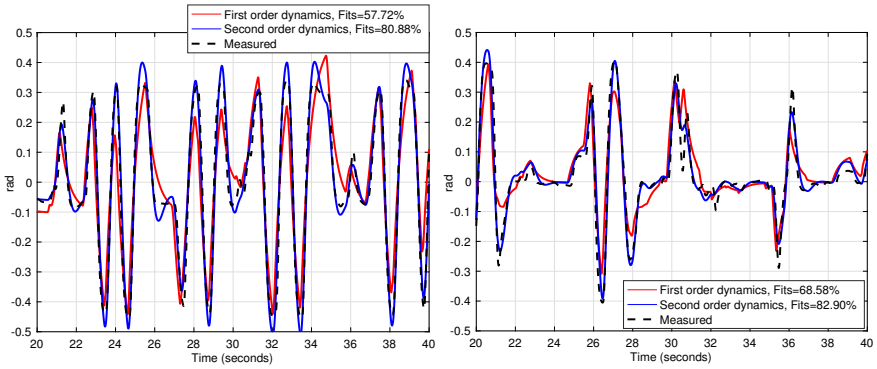
$y(s)_\phi$ and $u(s)_\phi$ are IMU measurement and input commands in continuous-time space. The time constants for roll and pitch are, $\tau_\phi = 0.472$, $\tau_\theta = 0.472$ and DC gains are $k_\phi = 1.673, k_\theta = 1.575$.

The identified first order dynamic models in continuous time space are discretized in MPC and will be addressed in the next section 3.3. We also performed

second order dynamic system identification exploited by disturbances observer, and the dynamic models are

$$\frac{y(s)_\phi}{u(s)_\phi} = \frac{26.37}{s^2+5.32s+27.04}, \frac{y(s)_\theta}{u(s)_\theta} = \frac{28.86}{s^2+6.00s+27.45}$$

Their gain, k_ϕ, damping, ζ_ϕ, and natural frequency, ω_ϕ are presented in Table 2. Fig. 4 shows measured attitude, estimated first and second order dynamics. The models fit close to the measurement (dotted line), and they are utilized by controllers presented in next section 3.3.



(a) (b)

Fig. 4 Measured and predicted pitch (a) and roll (b) angles for manual flight. The black denotes the measured angle, and the blue represents the model response.

	ϕ	θ	$\dot{\psi}$	$\dot{z}$
1st order	$\tau_\phi = 0.472$	$\tau_\theta = 0.472$	$\tau_{\dot{\psi}} = 0.161$	$\tau_{\dot{z}} = 0.334$
	$k_\phi = 1.673$	$k_\theta = 1.575$	$k_{\dot{\psi}} = 1.057$	$k_{\dot{z}} = 1.118$
2nd order	$k_\phi = 0.975$	$k_\theta = 1.052$	$k_{\dot{\psi}} = 1.079$	$k_{\dot{z}} = 1.024$
	$\zeta_\phi = 0.512$	$\zeta_\theta = 0.573$	$\zeta_{\dot{\psi}} = 1.898$	$\zeta_{\dot{z}} = 0.718$
	$\omega_\phi = 5.200$	$\omega_\theta = 5.239$	$\omega_{\dot{\psi}} = 23.448$	$\omega_{\dot{z}} = 4.985$

Table 2 Matrice 100 identified dynamics summary

3.2.3 Yaw and height dynamics

The input commands of yaw and height are rates, $u_{\dot{\psi}}, u_{\dot{z}}$, and the desired references are orientation, ψ^* and position, z^*. This implies there are controllers that track the desired yaw rate, $\dot{\psi}^*$, and height velocity, $\dot{z}^*$. Their first order dynamics are

$$\frac{y(s)_{\dot{\psi}}}{u(s)_{\dot{\psi}}} = \frac{5.642}{s+5.268}, \frac{y(s)_{\dot{z}}}{u(s)_{\dot{z}}} = \frac{3.342}{s+2.99}$$

Similarly, the second order dynamics for $\dot{\psi}$ and $\dot{z}$ are identified as

$$\frac{y(s)_{\dot{\psi}}}{u(s)_{\dot{\psi}}} = \frac{593.3}{s^2+89.0s+549}, \frac{y(s)_{\dot{z}}}{u(s)_{\dot{z}}} = \frac{25.43}{s^2+7.16s+24.8}$$

$y(s)_{\dot{\psi}}$ is obtained from the built-in IMU, gyro measurement along the z-axis, and $y(s)_{\dot{z}}$ is provided by a motion capture device. It is also feasible to identify the height dynamics by utilizing vertical velocity estimation from N1 flight controller.

3.3 Linear-MPC for horizontal control

The control of the lateral motion of the vehicle is based on a Linear Model Predictive Control (MPC) [14]. The vehicle dynamics are linearized around the hovering condition. We define the state vector to be $x = \left(\mathrm{x}, \mathrm{y}, v_{\mathrm{x}}, v_{\mathrm{y}}, {}^{\mathcal{W}}\phi, {}^{\mathcal{W}}\theta\right)$ and the control input vector to be $u = \left({}^{\mathcal{W}}u_{\phi}, {}^{\mathcal{W}}u_{\theta}\right)$. We also define the reference state sequence as $X_{\mathrm{ref}}^T = \left[x_{\mathrm{ref:0}}^T, \ldots, x_{\mathrm{ref:N}}^T\right]^T$, the control input sequence as $U = \left[u_0^T, \ldots, u_{\mathrm{N\text{-}1}}^T\right]^T$, and the steady state input sequence $U_{\mathrm{ref}} = \left[u_{\mathrm{ref:0}}^T, \ldots, u_{\mathrm{ref:N\text{-}1}}^T\right]^T$. $x_{\mathrm{ref:k}}, u_{\mathrm{ref:k}}$ are the $\mathrm{k_{th}}$ reference state and control input. Every time step, the following optimization problem is solved:

$$\begin{aligned}
\min_{U,X} \sum_{k=0}^{N-1} \Big(& \left(x_{\mathrm{k}} - x_{\mathrm{ref:k}}\right)^T Q_{\mathrm{x}} \left(x_{\mathrm{k}} - x_{\mathrm{ref:k}}\right) \\
& + \left(u_{\mathrm{k}} - u_{\mathrm{ref:k}}\right)^T R_{\mathrm{u}} \left(u_{\mathrm{k}} - u_{\mathrm{ref:k}}\right) \\
& + \left(u_{\mathrm{k}} - u_{\mathrm{k\text{-}1}}\right)^T R_{\Delta} \left(u_{\mathrm{k}} - u_{\mathrm{k\text{-}1}}\right) \Big) \\
& + \left(x_{\mathrm{N}} - x_{\mathrm{ref:N}}\right)^T P \left(x_{\mathrm{N}} - x_{\mathrm{ref:N}}\right) \\
\text{subject to} \quad & x_{\mathrm{k+1}} = A x_{\mathrm{k}} + B u_{\mathrm{k}} + B_{\mathrm{d}} d_{\mathrm{k}}; \\
& d_{\mathrm{k+1}} = d_{\mathrm{k}}, \quad \mathrm{k} = 0, \ldots, N-1 \\
& u_{\mathrm{k}} \in \mathbb{U} \\
& x_0 = x\left(t_0\right), \quad d_0 = d\left(t_0\right).
\end{aligned} \tag{2}$$

where $Q_{\mathrm{x}} \succeq 0$ is the penalty on the state error, $R_{\mathrm{u}} \succ 0$ is the penalty on control input error, $R_{\Delta} \succeq 0$ is a penalty on the control change rate and P is the terminal state error penalty. The $\succeq$ operator denotes positive definiteness of a matrix[5]. d_{k} is the estimated external disturbances. Note that the attitude angles ϕ, θ are rotated into the inertial frame to get rid of the vehicle heading ψ.

A high-performance solver has been generated to solve the optimization problem (2) using the FORCES[6] framework. The solver is running in real-time for a prediction horizon of $N = 20$ steps. Moreover, to achieve an offset-free tracking, the external disturbances d_k has to be estimated and provided to the controller each time step. These disturbances include external forces (the wind for instance) and also a modeling error. The disturbances are estimated using an augmented Extended Kalman Filter (EKF) based on the second order dynamics model identified from the previous section. For the remaining axes control (i.e., height and yaw), we use standard PID controllers.

[5] https://en.wikipedia.org/wiki/Positive-definite_matrix

[6] http://embotech.com/FORCES-Pro

4 Experimental results

In this section, we present implementation details; hardware and software setup and control performance evaluation through experiments.

4.1 Hardware Setup

Matrice 100 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. A Vicon motion capture system consisting of 6 IR cameras provides 6 DOF pose of the quadcopter target at 100 Hz, used for control. The quadcopter is also equipped with a flight controller, N1, embedded an onboard IMU which provides vehicle orientation, acceleration, and angular velocity at 100 Hz to the computer via 921,600 bps USB-to-serial communication. Control commands are also transmitted at 100 Hz through the serial bus.

The total system mass is 3.3 kg and WiFi is used for communicating with the quadcopter using a ground control laptop via ROS and a customized onboard SDK ROS interface. The ground station, Vicon server, and onboard computer are time synchronized via chrony time sync software package. The vehicle carries 0.92 kg payload with the onboard computer, and a gimbal camera. 6 cells LiPo battery, 22.2 V, 4500 mAh powers the vehicle and the total flight time is around 14 mins with a small angle of attack $\approx \pm 20^{\circ}$.

4.2 Software Setup

We integrate the system using ROS as shown in Fig. 5. Each box represents a ROS node and runs at 100 Hz. The Vicon server publishes position, orientation, translational and angular velocity as denoted $[p, q, \dot{p}, \dot{q}]$ using ros_vrpn_client. The data is subscribed by the Multi-Sensor Fusion (MSF) framework [6] to filter noisy measurement and to compensate for a delay. The ground station sets either a goal pose as denoted p^*, position, and $q^*]$, orientation or N sequences, $[p^*_{1:\mathrm{N}}, q^*_{1:\mathrm{N}}]$, planned by the trajectory generator [11].

The SDK runs on the onboard computer and receives IMU measurements, $[\Phi, \dot{\Phi}, {}^{\mathcal{B}}a]$, orientation, angular velocity, acceleration in $\mathcal{B}$ coordinate from the N1 flight controller. It also sends the calculated control commands to the attitude controller.

4.3 Experiments Setup

For control performance evaluation, we conduct real-world experiment: hovering, step response, and trajectory following. To demonstrate, the robustness of the controllers, we generate wind disturbances using a fan that has 260 W and 300 $\mathrm{m}^3/\mathrm{min}$ air flow. This produces 11-11.5 m/s disturbance at the hovering position measured by an anemometer. Each task has two results, i.e., with/without wind disturbances.

We use root-mean-square (RMS) error metric between the reference and actual position and orientation measured by a motion capture device for performance evaluation. Euclidian distance is used for 3D pose RMS error calculation. Table 3 summarizes experimental results.

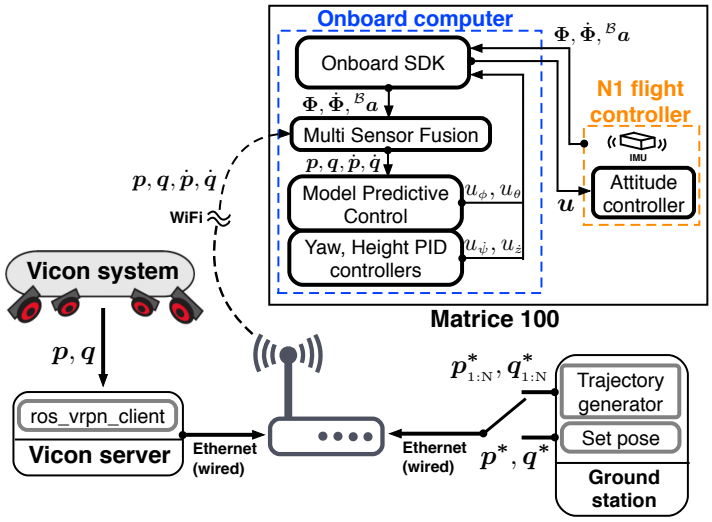


Fig. 5 Software packages implementation using ROS.

4.4 Control performance evaluation

We present quantitative control performance evaluation using RMS error while performing 3 tasks, i.e., hovering, step response, and trajectory following, and qualitative results for step response and trajectory following.

	Hovering		**Step response**		**Trj. following**		Unit
Pose	**0.039**	0.041	0.394	0.266	0.080	0.103	m
x	**0.021**	0.027	0.259	0.127	0.058	0.066	m
y	0.016	**0.015**	0.295	0.226	0.043	0.059	m
z	0.029	**0.026**	0.034	0.059	0.035	0.053	m
roll	0.392	1.044	—	—	—		deg
pitch	0.618	0.697	—	—	—		deg
yaw	**1.087**	1.844	1.141	2.165	1.539	2.876	deg
Duration	15-75	15-75	20-120	20-120	30-80	20-70	s
Wind	—	11-11.5	—	11-11.5	—	11-11.5	m/s

Table 3 Control performance summary

4.4.1 Experiments results for hovering

Fig. 6 shows hovering results (position and orientation) without any wind disturbances (a), (b) and with disturbances (c), (d). Noticeable areas are magnified due to the small scale of plots. We can clearly see the presence of wind disturbances affect to control performance. Especially, the variation in yaw and attitude are significant since a fan is located at South-East of the vehicle. As shown in Table 3, we achieve competitive results while hovering. Interestingly, the position errors for x, y, and z are consistent even with wind disturbances, 11-11.5 m/s. The force acting on the platform in the wind is too small to push the 3.3 kg flying platform.

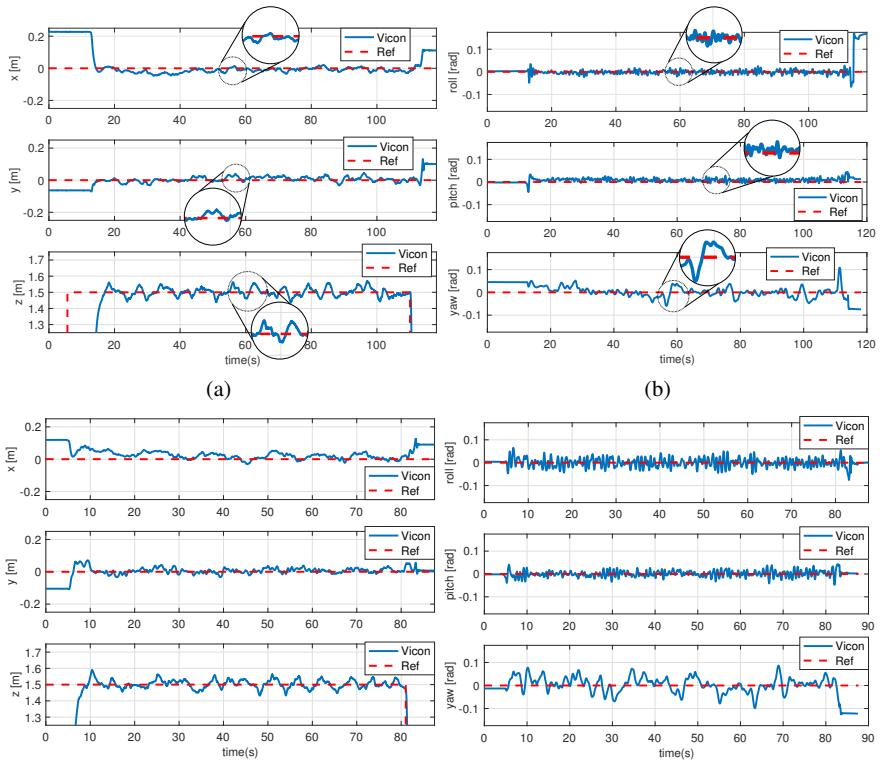


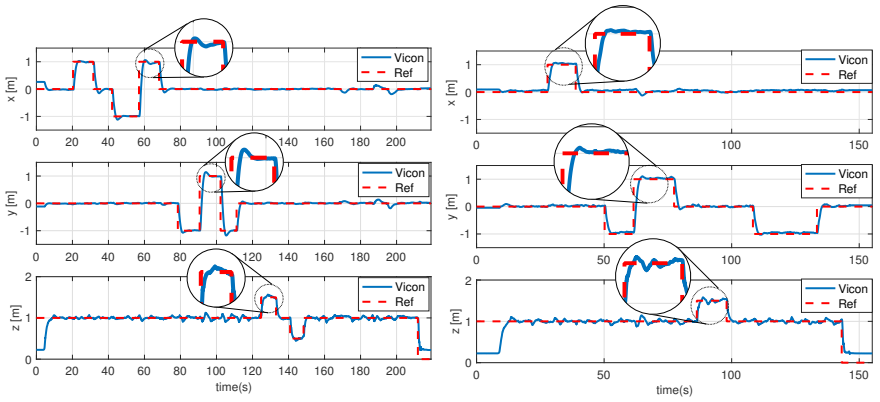
Fig. 6 Hovering control performance without/with wind disturbances.

4.4.2 Experiments results for step response

As oppose to the advantage of strong resistance to external disturbances, the downside for the heavy platform is a slower response. Fig. 7 shows step response plots without wind disturbances (a), with wind (b). A goal position is manually chosen to excite all axises. The peaks in roll and pitch are caused by tilting toward the direction of maneuver, so we ignore them in RMS calculation. The results from Table 3 show a large control error in both x and y. Slow response causes accumulating error while the vehicle reaches to a reference goal. Note that the RMS error of x-axis in windy condition (i.e., 0.127 m) is much smaller than without wind (0.259 m). This is mainly due to the vehicle travels only 2 m for the former whereas it flies 6 m for the latter.

4.4.3 Experimental results for trajectory following

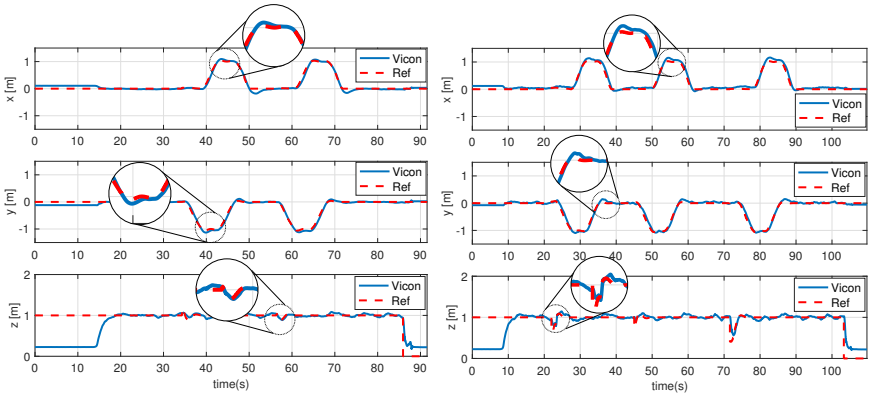
We use [2] to generate a smooth polynomial reference trajectory as shown in Fig. 9. Even though hovering and step response tasks explicitly demonstrate control performance and essential functionalities for VTOL MAVs, trajectory following is also a significant task for many robotic applications such as obstacle avoidance, and path



(a) (b)

Fig. 7 Step response control performance without (a) and with (b) wind disturbances.

planning. Fig. 8 shows trajectory following results without wind disturbances (a) and with the wind (b). It can be seen that the platform tracks the reference well in both conditions. RMS errors are also presented in Table 3.



(a) (b)

Fig. 8 Trajectory following control performance without (a) and with (b) wind disturbances.

4.4.4 Qualitative results

We present two qualitative results for trajectory following and step response. Fig. 9 illustrates the planned trajectory (red) and the vehicle position (blue) obtained from a motion capture device. The left column is without the wind, and the right is in windy condition. Note that a fan is located around 3 m away from the hovering position along the South-East direction as illustrated. Each row is top and side views. It is clearly seen that the trajectory is shifted to the wind direction (positive x and y-direction). Fig. 10 shows motions of step response (a) and trajectory following (b). For the step response, it can be seen that the vehicle builds up moments by tilting toward the goal direction and decelerates by tilting into the opposite direction

when it approaches. For the trajectory following, the vehicle accurately follows 1×1 square shown in Fig. 10(b).

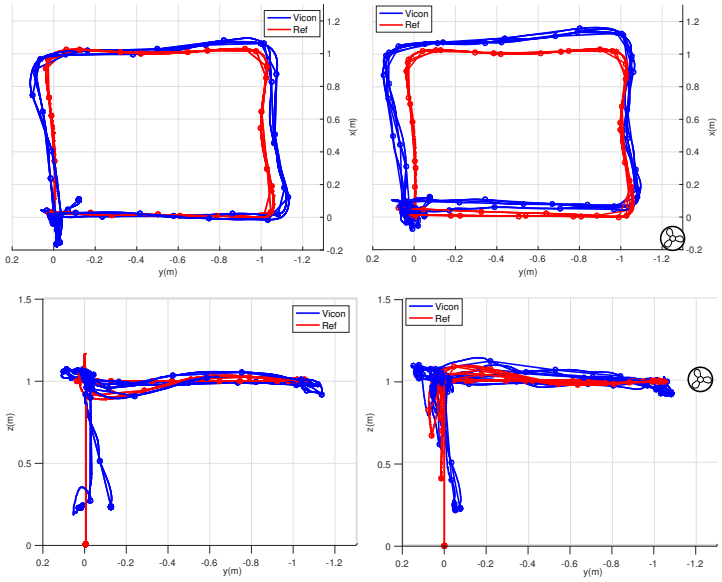


Fig. 9 3D pose during trajectory following without (left column) and with (right column) wind disturbances.

(a) (b)

Fig. 10 Vehicle motion while step response (a) and trajectory following (b).

5 Conclusions

We have presented control performance of a cost-effect VTOL MAV platform using classic system identification techniques and the-state-of-the-art MPC controller. The essential information for developing the VTOL MAV robotic platform such as attitude dynamics are addressed. The applied controller performance are evaluated through experiments while executing hovering, step response and trajectory following. Using this platform has many advantages such its low-cost, high payload, ease of use and the ready availability of replacement parts, user-friendly interface, powerful SDK, and a large user community. Our experiences and findings are returned to the community through open documentation and software packages to support

researchers having their high-performance MAV platform for diverse field-service aerial robot applications.

Acknowledgement
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644227 and from the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0029.

References

1. Sa I. and Corke P. (2012) Vertical Infrastructure Inspection using a Quadcopter and Shared Autonomy Control. The International Conference on Field and Service Robotics.
2. Burri, M., Oleynikova, H., Achtelik, M., and Siegwart, R. (2015) Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments. IEEE/RSJ International Conference on Intelligent Robots and Systems.
3. Mellinger, D., Lindsey, Q., Shomin, M., and Kumar, V. (2011) Design, modeling, estimation and control for aerial grasping and manipulation. IEEE/RSJ International Conference on Intelligent Robots and Systems.
4. Zhang, C., and Kovacs, J. (2012) The application of small unmanned aerial systems for precision agriculture: a review. Precision agriculture.
5. Achtelik M., Weiss A., and Siegwart R. (2011) Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments. Proceedings of the IEEE International Conference on Robotics and Automation.
6. Weiss, S., Scaramuzza, D., and Siegwart, R. (2011) Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments. Journal of Field Robotics.
7. Bouabdallah, S. (2007) Design and control of quadrotors with application to autonomous flying. Ecole Polytechnique Federale de Lausanne, PhD Thesis.
8. Mahony, R., Kumar, V., and Corke, P. (2012) Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. IEEE Robotics and Automation Magazine.
9. Pounds, P., Mahony, R., and Corke, P. (2010) Modelling and control of a large quadrotor robot. Control Engineering Practice.
10. Hoffmann, M., Waslander, S., and Tomlin, C. (2008) Quadrotor helicopter trajectory tracking control. AIAA guidance, navigation and control conference and exhibit.
11. Burri, M., Nikolic, J., Oleynikova, H., Achtelik, M., and Siegwart, R. (2016) Maximum likelihood parameter identification for MAVs. IEEE International Conference on Robotics and Automation.
12. Sa I, and Corke P. (2012) System Identification, Estimation and Control for a Cost Effective Open-Source Quadcopter. IEEE International Conference on Robotics and Automation.
13. Tischler, M. and Remple, R. (2006) Aircraft and rotorcraft system identification. AIAA education series.
14. Mina K., Thomas S., Kostas A., and Roland S. (2016) Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System. Springer Press (to appear).
15. Marija Popovic, Gregory Hitz, Juan Nieto, Inkyu Sa, Roland Siegwart, and Enric Galceran (2016) Online Informative Path Planning for Active Classification Using UAVs. IEEE International Conference on Robotics and Automation.