Journal of ELECTRICAL ENGINEERING, VOL 70 (2019), NO1, 3–15

sciendo

PAPERS

Control of a small quadrotor for swarm operation

Adam Trizuljak*, Frantiek Duchoň*, Jozef Rodina*, Andrej Babinec*, Martin Dekan*, Roman Mykhailyshyn**

Small quadrotors, or so-called nanoquads, are widely available, typically have small take-off mass (between 12–50 g), and a flight time of about 5–10 minutes. The aim of this article is the proposal of control and development of the basic infrastructure for controlling a swarm nanoquads from an external computer and obtaining measurements from an onboard sensor. Control of nanoquad attitude and position is proposed and control allocation problem is addressed. Additionally, landing and collision detection is implemented using external disturbance force estimation. Results of the proposed control methods are verified in 4 scenarios: hover flight, manual control, step response, and collision and landing detection.

K e y w o r d s: miniature quadrotor, nanoquad control, UAV, swarm

1 Introduction

Advancements in Unmanned Aerial Vehicle (UAV) technology have allowed the construction of ever-smaller quadrotors with increasing onboard computational power and flight performance. In the recent years, several new quadrotors have been developed specifically for research purposes, such as in [1–3] or [19], ranging in size and onboard computational power. Lately, several commercial and/or open-source projects have also become available, such as the Crazyflie [4] or Phenox [5], which are highly suitable for research work.

Previous quadrotor swarm projects have so far featured a centralized, non-autonomous approach, where a single computer (ground station) is running position controllers for all quadrotors in the swarm [1, 6, 21]. As of the time of writing (2016), a decentralized approach, where each of the quadrotors runs its own position controller has been so far relatively uncommon and related research only started to appear very recently [3, 7]. In [7] authors implemented a swarm of 49 Crazyflie 2.0 quadcopters with on-board position control and centralized trajectory generation. They found the same limitations mainly regarding motion tracking and radio packet throughput. This work has been carried out simultaneously and independently from us between years 2015 – 2016 and still arrived at similar conclusions, which generally confirms the chosen approach towards solving this problem.

The goal of our work was to develop the basic infrastructure controlling of swarm of nanoquads from an external computer and obtaining measurements from an onboard sensor. The nanoquads must be cheap, easily procurable and extendable, must have sufficient payload for sensors, and possibility to control it externally.

In this paper, an approach where each of the quadrotors is semi-autonomous is implemented – it runs its own position controller in the on-board firmware, based on current pose data and commands received from the ground station, therefore the task of position control becomes decentralized. The ground station is responsible only for gathering of pose data and flight trajectory computation and generation. It then commands the individual quadrotors with waypoints (position, velocity, acceleration.) based on the generated trajectory. Furthermore, each quadrotor is equipped with a sensor to be capable of remote sensing and survey. The development of a quaternion-based attitude controller with torque disturbance observer is shown, replacing the default Euler angle controller. A position controller along with external disturbance force observer is implemented, which is also utilized to detect collisions and landing of the quadrotor.

2 Position tracking

Estimating the pose of a quadcopter in space is essential for any UAV. In many cases, this is accomplished by means of a motion capture system. It typically consists of several infrared cameras mounted statically in a room and a dedicated computer, which processes the data from all cameras and runs pose estimation algorithms. User interface software is provided for adjusting the system parameters, managing the tracked objects and for viewing, recording and playback of the pose data. In our experiments a Vicon Bonita B10 system consisting of 14 cameras with a total tracking volume of $6 \times 10 \times 3.2$ meters was used. The pose estimation runs at 250 Hz and the object pose data is available via a network packet stream.

* Slovak University of Technology in Bratislava, Slovakia, frantisek.duchon@stuba.sk, ** Ternopil Ivan Puluj National Technical University, Ukraine

DOI: 10.2478/jee-2019–0001, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

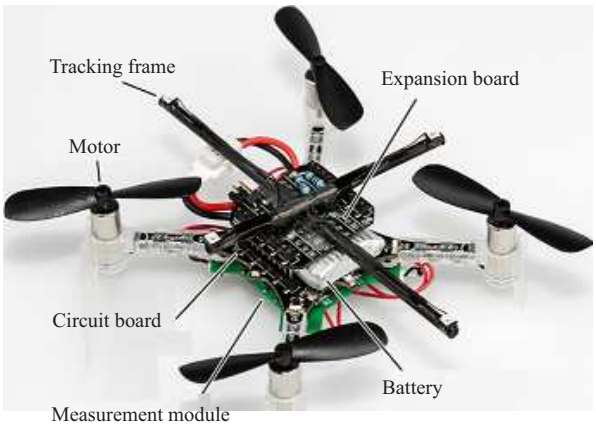

Fig. 1. Overview of the on crazyflie 2.0 platform with our IR LED tracking body

A motion capture system such as Vicon or ART typically uses Infrared (IR) cameras equipped with IR strobe units. The strobe pulse is synchronized with the camera shutter to illuminate the scene while an image is being captured. Most motion capture systems use ball-shaped tracking markers coated in retro-reflective material. These are often referred to as passive markers, since they do not actively emit light. When illuminated by IR light from the strobe, these markers appear as very bright, well-defined blobs in the camera picture. Another approach is to use infrared light emitting diodes (LED). The emitted light is detected the same way as with passive markers, however due to the much smaller size of a LED the maximum detection distance is decreased. The LED also has a narrower angle of view (*ie* 120°) when compared to a passive marker, therefore the possible detection angle is decreased. LED-based markers are referred to as active markers.

The system then runs blob detection to determine the 2D position of the center of each detected blob within the camera frame and uses data from cameras which see the marker to triangulate the overall 3D position of the marker. To accurately estimate the position and orientation of an object in 3D space, multiple markers (typically 4-5) have to be used. For this purpose, a tracking body is composed of multiple markers at defined distances from each other. There are several requirements and constrains for the shape of the body:

- The shape of the body must not be symmetrical.
- The shape must be very rigid and remain mechanically stable.
- The individual markers must be separated by a certain minimum distance. This distance is proportional to the marker diameter, the camera resolution, the maximum distance from the camera and the strobe intensity.
- For tracking multiple objects, each object requires a separate and unique tracking body.

In a traditional system, the markers have to be large enough to be reliably and accurately detected by the cameras. For optimal results, it is generally recommended to use as large tracking markers as the tracked object allows or can carry. Commonly available markers range from roughly 15 mm to 30 mm in diameter. However, small quadcopters such as the Crazyflie with their small size and lift capability are simply not capable of carrying these. This issue is usually solved by using a very small markers, as can be seen *eg* in [1] or [2]. Markers roughly 10 mm in diameter were used on a platform which is much larger than the Crazyflie and therefore the markers could be placed in a way that they do not get overlaid by the quadcopter body.

In order to overcome this problem, the first attempt was to use active markers based on infrared LEDs. The Vishay VSMF4710 LED ($\lambda = 870$ nm, 120° viewing angle, $I_f = 100$ mA) [8] was chosen as our starting point. 4 LEDs connected in two parallel strings of two LEDs with a 10 Ω current limiting resistor were used. The strings are connected to the positive pole of the main 3.7 V Lithium-polymer (LiPo) battery via a Crazyflie expansion/breakout board. As of now, the LEDs are operated continuously. The tracking body is built from 3 mm diameter carbon fiber tubes glued to the expansion board to form an asymmetrical X-shape (Fig. 1), and a LED is glued on each end of the tubes. The assembly weighs 3.3 g and is connected to the Crazyflie as a regular expansion board. In comparison to mounting the LEDs directly onto the Crazyflie frame, this construction improves visibility of the markers and provides more freedom to create unique combinations of marker positions.

However, using LED-based markers does have some disadvantages. As discussed before, the smaller apparent blob size decreases the maximum detection distance from the camera and may result in a less accurate position estimate. The Vicon system considers blobs under a certain configurable size as invalid to avoid false detections caused *eg* by a metallic reflection. The second issue is the power consumption. Since each LED has forward voltage $V_\mathrm{f} = 1.5$ V and drive current $I_\mathrm{f} = 100$ mA, and the Crazyflie has a battery with only 240 mAh capacity, this additional power drain can significantly reduce the flight time.

This problem could be solved by modulating the LEDs at the motion capture system update frequency (*ie* 250 Hz) and to have them turned on only for the duration of frame capture, therefore lowering the duty cycle (and power consumption) significantly. The simplest solution is to detect the IR strobe pulse with a photodiode and use a comparator to modulate the marker LEDs. A simple circuit similar to [9] could be used for this purpose.

Given the problems associated with LED-based markers, it was also attempted to use passive markers for motion tracking. Hemispheres with diameter of 7 mm coated in retro-reflective material were used, creating tracking bodies composed of 4 or 5 of these markers. An example configuration can be seen in Fig. 2. The markers were glued directly onto the quadrotor frame. The advantage of the passive markers is the reduced weight and zero

Fig. 2. Crazyflie 2.0 with passive markers and a protective shroud

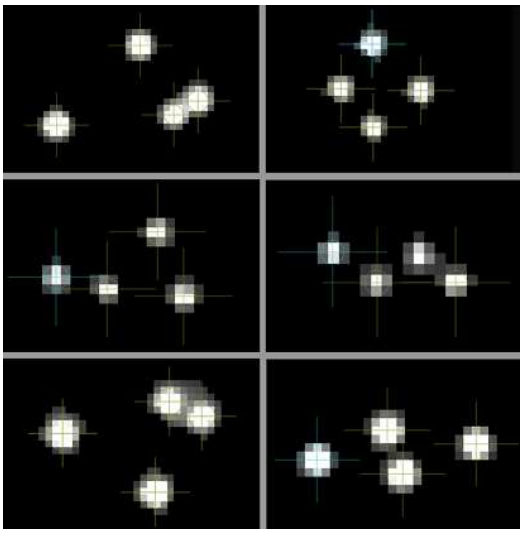

Fig. 3. A comparison of camera images from Vicon motion capture system: left – an off-the shelf passive marker, right – active LED marker; the individual rows represent views from three different cameras at various distances from the markers, the faint blue cross represents a selected marker

power draw, thanks to which the flight time is increased. The construction is also greatly simplified. However, compared to the previously proposed LED-based tracking body, it is more difficult to create multiple unique tracking bodies, because the quadrotor frame offers relatively few locations, where a marker can be placed. Also, markers of this size are approaching the limit of what can be reliably detected by our relatively large volume motion tracking system. Lastly, adding a protective shroud such as in Figure 2 can occlude some of the markers and make the tracking less reliable. This shroud was designed to enable collision detection, to provide protection to the quadrotors and to improve human safety.

The problem of creating enough unique marker combinations has been approached differently in [7]. Authors used identical tracking bodies for all of their quadrotors and implemented their own tracking software using an Iterative Closest Point (ICP) algorithm able to handle the identical bodies. With each new pose measurement, it takes the raw point cloud from the motion capture system as the input and for each frame uses ICP to register the last known pose of the quadrotor to the current point cloud. This approach is capable of estimating the quadrotors pose at 75 Hz with 49 quadrotors flying simultaneously. It successfully solves the problem of having enough unique tracking bodies, at the cost of increased implementation complexity.

In the end, it was found that the passive markers are more advantageous over the LED based markers and it was decided to use them for experiments (Figure 3). However, if it is desirable to expand the quadrotor swarm to a higher count, the approach to tracking will have to adapt in order to create additional unique tracking bodies.

3 Control

The required properties of nanoquads are low price, easy procurement and extension, sufficient payload, and possible external control. After evaluating all available platforms, it was decided to use the Crazyflie 2.0. The original Crazyflie 2.0 firmware utilizes an Euler angle-based proportional-integral-derivative (PID) attitude controller implemented as a cascade controller. The inner loop is comprised of a P angular rate controller, and the outer loop is a PI Euler angle controller.

Euler angles-based attitude controllers suffer from rotational singularities. To address this issue, a quaternion-based proportional-derivative (PD) attitude controller with an angular acceleration-based disturbance observer (DO) was implemented. The attitude controller design follows the approach presented in [18]. In addition, a second-order quaternion pre-filter with angular velocity feed-forward was implemented. This implementation is designed to be an "in-place" replacement of the old controller, maintaining backwards compatibility with the rest of the firmware structure.

The state of the quadrotor in free flight is described by the position $\boldsymbol{p} = [x \quad y \quad z]^\top$ and attitude unit quaternion $\boldsymbol{q}_{bi}$ (Figs. 4, 5). The attitude control torques $\boldsymbol{\tau} = [\tau_x \ \tau_y \ \tau_z]^\top$ are exerted by the four propellers, along with a total thrust force $\boldsymbol{f}$. The model uncertainties and external disturbances are represented by the terms $\boldsymbol{f}_e$ and $\boldsymbol{d}$. The quadrotor dynamics can be then described using the translational acceleration $\ddot{p}$ and the rotational acceleration $\dot{\omega}$

$$m\ddot{\boldsymbol{p}} = m \geq_3 + \boldsymbol{R}_{ib}\boldsymbol{f} + \boldsymbol{f}_e \,, \tag{1}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = (\boldsymbol{J}\boldsymbol{\omega}) \times \boldsymbol{\omega} + \boldsymbol{\tau} + \boldsymbol{d} \tag{2}$$

where m is the quadrotor mass and moment of inertia, $\boldsymbol{g}$ is the gravitational acceleration vector and $\boldsymbol{e}_3 = [0 \ 0 \ 1]^\top$ is the unity vector in the direction of the inertial z - axis. The rotation matrix $\boldsymbol{R}_{ib}$ describes the quadrotors current attitude in the inertial reference frame and can be calculated from $\boldsymbol{q}_{bi}$. $\boldsymbol{J}$ denotes a 3×3 matrix with values

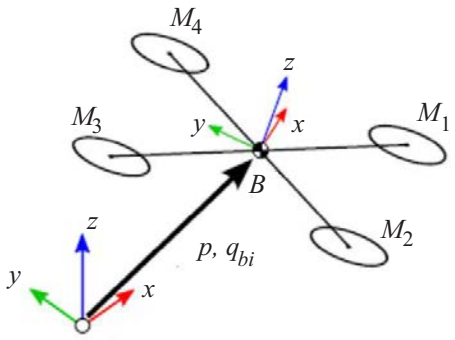

Fig. 4. Pose of the quadrotor body frame B in inertial frame of reference $\boldsymbol{I}_V$

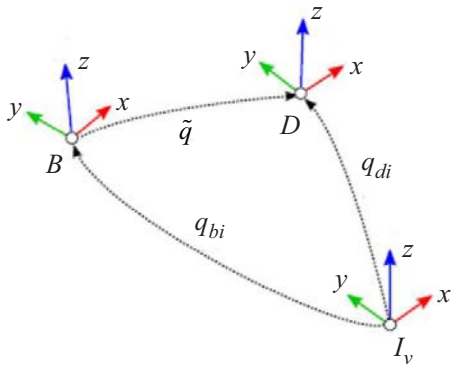

Fig. 5. Quaternion transformations between inertial, desired and body frame

of mass moment of inertia around principal rotational axes of the quadrotor on the diagonal

$$\boldsymbol{J} = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}. \tag{3}$$

Values of $\boldsymbol{J}$ were obtained using the simplified method described in [11] and [20]. In this paper the following quaternion notation is used

$$\boldsymbol{q} = [\,\boldsymbol{\eta} \quad \boldsymbol{\epsilon}\,]^\top = [\,w \quad x \quad y \quad z\,]^\top \tag{4}$$

where $\boldsymbol{\eta} = w$ is the scalar part and $\boldsymbol{\epsilon} = [x\ y\ z]^\top$ is the vector part.

3.1 Attitude control

The orientation of the quadcopter body-fixed frame B in the Vicon motion capture system reference frame $\boldsymbol{I}_V$ is described by a unit quaternion $\boldsymbol{q}_{bi}$. This quaternion is obtained from the on-board sensor fusion, which processes data from the on-board 6-axis inertial measurement unit (IMU) using the Mahony attitude and heading reference system (AHRS) algorithm [2, 12] at 500 Hz update rate. The on-board sensor fusion also incorporates basic signal conditioning, most importantly a low pass filter of the gyroscope data. The quaternion used internally by the AHRS algorithm is also updated using pose measurements from the Vicon motion tracking system which enhances attitude estimation accuracy and eliminates the effect of yaw drift caused by gyroscope bias. The coordinate frame of the quadrotor (Fig. 6) is chosen to match the coordinate frame of the IMU. The orientation of the desired frame D is described by the unit quaternion $\boldsymbol{q}_{di}$.

Figure 6 shows the high-level block diagram of the Crazyflie control system. The desired quaternion $\boldsymbol{q}_{di}$. is obtained either from the remote-control input, or from the position controller output (represented by switches in the block diagram). The remote control input consists of desired Euler angles $[\Phi(^\circ), \Theta(^\circ), \Psi(^\circ \mathrm{s}^{-1})]^\top$ and Thrust T, a 16-bit unsigned integer representing total thrust between 0 N and cca 0.6 N [14]. To maintain backwards compatibility with the rest of the system (*ie* the Crazyflie Client software and the Crazyflie smartphone application), which relies on sending/receiving these commands in degrees, the Euler angles need to be first converted into radians, before being transformed into the quaternion $\boldsymbol{q}_{di}$ [15].

The goal of the attitude controller is to align the body frame B with the desired frame D. The quaternion $\widetilde{\boldsymbol{q}}$ describes the transformation from coordinate frame B to the coordinate frame D, or in other words the quaternion error between B and D, such $\widetilde{\boldsymbol{q}} = \boldsymbol{q}_{\mathrm{db}} = \boldsymbol{q}_{\mathrm{di}} \otimes \boldsymbol{q}_{\mathrm{bi}}$.

The error quaternion is calculated as

$$\widetilde{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{\eta}_{\mathrm{di}} & \boldsymbol{\epsilon}_{\mathrm{di}}^\top \\ -\boldsymbol{\epsilon}_{\mathrm{di}} & \boldsymbol{\eta}_{\mathrm{di}}\boldsymbol{I} - S(\boldsymbol{\epsilon}_{\mathrm{di}}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_{\mathrm{bi}} \\ \boldsymbol{\epsilon}_{\mathrm{bi}} \end{bmatrix} \tag{5}$$

where $S(\boldsymbol{q})$ is a 3×3 skew-symmetric operator matrix. The desired torque $\boldsymbol{\tau}$ is calculated by the control law,

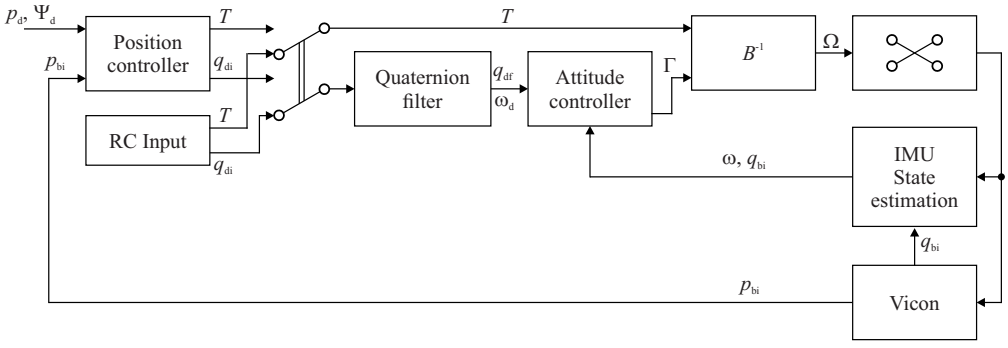

Fig. 6. High-level block diagram of the crazyflie control system

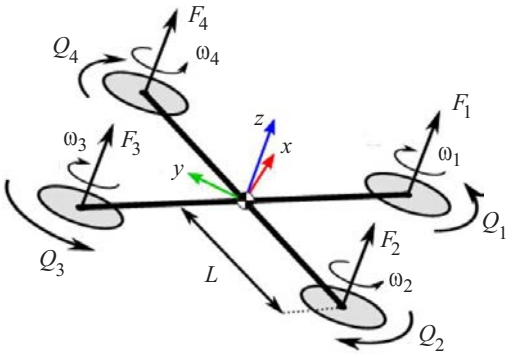

Fig. 7. Motors $M_{1..4}$ with propellers spinning at angular velocities $\boldsymbol{\omega}_{1..4}$ generate thrust forces $F_{1..4}$ and propeller drag torques $Q_{1..4}$

which is defined as

$$\boldsymbol{\tau} = [\,\tau_x \quad \tau_y \quad \tau_z\,]^\top = \boldsymbol{J}(-\boldsymbol{K}_d\boldsymbol{\omega} - 2\widetilde{\boldsymbol{\eta}}\boldsymbol{K}_p\widetilde{\boldsymbol{\epsilon}}) \tag{6}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are the proportional and derivative gain matrices, respectivel $\widetilde{\boldsymbol{q}} = [\widetilde{\eta}\ \ \widetilde{\boldsymbol{\epsilon}}]^\top$ is the quaternion orientation error obtained from (5) and $\boldsymbol{\omega}$ is the vector of angular velocities of the quadrotor obtained from the IMU.

Furthermore, the second-order quaternion filter as proposed in [16] was implemented. The filter takes the desired quaternion $\boldsymbol{q}_{di}(t)$ as the input and produces the filtered desired quaternion $\boldsymbol{q}_{df}(t)$ and the corresponding desired angular velocity $\boldsymbol{\omega}_{df}(t)$ as the output. The filter is defined as

$$\widetilde{\boldsymbol{q}}_f = [\widetilde{\boldsymbol{\eta}}_f\ \ \widetilde{\boldsymbol{\epsilon}}_f]^\top = \boldsymbol{q}_{\mathrm{d}i} \otimes \boldsymbol{q}_{\mathrm{d}f}\,, \tag{7}$$

$$\dot{\boldsymbol{\omega}}_{\mathrm{df}} = \alpha(-2\widetilde{\boldsymbol{\eta}}_{\mathrm{f}}\widetilde{\boldsymbol{\epsilon}}_{\mathrm{f}} - \boldsymbol{\omega}_{\mathrm{df}})\,, \tag{8}$$

$$\widetilde{\boldsymbol{q}}_{\mathrm{d}f} = \frac{1}{2}\widetilde{\boldsymbol{q}}_f \otimes \boldsymbol{\omega}_{\mathrm{d}f} \tag{9}$$

where α is the filter gain. The multiplication in (9) is quaternion multiplication, where $\boldsymbol{\omega}_{\mathrm{d}f}$ is treated as a vector quaternion. Filtered desired quaternion and the desired angular velocity into the control law are then incorporated

$$\boldsymbol{\tau} = \boldsymbol{J}(-\boldsymbol{K}_d(\boldsymbol{\omega} - \boldsymbol{\omega}_{\mathrm{d}f}) - 2\widetilde{\boldsymbol{\eta}}_{fi}\boldsymbol{K}_p\widetilde{\boldsymbol{\epsilon}}_{fi}) \tag{10}$$

where the quaternion error is now calculated as $\widetilde{\boldsymbol{q}}_{fi} = \boldsymbol{q}_{\mathrm{d}f} \times q_{\mathrm{b}i}$. A model-based disturbance observer (DO) is implemented in order to improve the attitude tracking performance. From the system model (2) the disturbance $\boldsymbol{d}$ of the control torque $\boldsymbol{\tau}$ can be expressed as

$$\boldsymbol{d} = \boldsymbol{J}\dot{\boldsymbol{\omega}} - (\boldsymbol{J}\boldsymbol{\omega}) \times \boldsymbol{\omega} - \boldsymbol{\tau}\,. \tag{11}$$

As our angular velocities are usually small and decoupled, the effect of the Coriolis term $(\boldsymbol{J}\boldsymbol{\omega}) \times \boldsymbol{\omega})$ can be neglected in the normal flight. Since the angular acceleration is obtained by differentiating the IMU gyroscope data and is thereby noisy, a simple low pass filter with the gain $K_{\mathrm{DO},a}$ is implemented and the total disturbance estimate $\widehat{\boldsymbol{d}}$ is integrated over time

$$\dot{\widehat{\boldsymbol{d}}} = \boldsymbol{K}_{\mathrm{DO},a}(\boldsymbol{d} - \widehat{\boldsymbol{d}})\,. \tag{12}$$

3.2 Control allocation

The control input $\boldsymbol{\Omega} = [\boldsymbol{\tau}_x\ \boldsymbol{\tau}_y\ \boldsymbol{\tau}_z\ T]^\top$ consists of the desired torques $\boldsymbol{\tau}_{x,\,y,\,z}$ (Nm) around the principal axes and the thrust force $T(N)$. It is needed to calculate the motor control input $\boldsymbol{u} = [T_1\ T_2\ T_3\ T_4]^\top$, T_i being the desired thrust force of the i-th motor. To determine the control allocation, the simplified quadrotor dynamics model depicted in Fig. 7 was used. Motors $M_{1\ldots4}$ with propellers are placed 90° apart in one plane at the distance L from the center of mass. Each propeller, spinning at an angular velocity $\boldsymbol{\omega}_i$ produces a thrust force $\boldsymbol{F}_i = k_F\boldsymbol{\omega}^2$ in the direction of the frame z axis and a propeller drag torque $Q_i = k_0\boldsymbol{\omega}^2$ with the direction opposite of $\boldsymbol{\omega}_i$. The coefficients k_F and k_Q are the propeller thrust and drag coefficients, respectively. The control allocation problem can subsequently be formulated as

$$\Omega = \boldsymbol{B}\boldsymbol{u} \tag{13}$$

where $\boldsymbol{B}$ is the control allocation matrix. Using the standard torque equation $\boldsymbol{\tau} = \boldsymbol{r} \times \boldsymbol{F}$, where $\boldsymbol{r}$ is the position vector and $\boldsymbol{F}$ is the force vector, the contribution of each motor to the overall torque around the x and y axes can be identified (14a,b). The torque around the z axis is the sum of individual rotor drag torques $\boldsymbol{Q}_i$ (14c). Total thrust force T is the sum of individual motor thrust forces T_i (14d).

$$\boldsymbol{\tau}_x = (-T_1 - T_2 + T_3 + T_4)L\cos\frac{\pi}{4} \tag{14a}$$

$$\boldsymbol{\tau}_y = (-T_1 + T_2 + T_3 - T_4)L\sin\frac{\pi}{4} \tag{14b}$$

$$\boldsymbol{\tau}_z = -Q_1 + Q_2 - Q_3 + Q_4 \tag{14c}$$

$$T = \sum_{i=1}^{4} T_i \tag{14d}$$

To obtain the matrix B, equations (14a–d) are expressed in matrix form

$$\Omega = \begin{bmatrix} -L\cos\frac{\pi}{4} & -L\cos\frac{\pi}{4} & L\cos\frac{\pi}{4} & L\cos\frac{\pi}{4} \\ -L\sin\frac{\pi}{4} & L\sin\frac{\pi}{4} & L\sin\frac{\pi}{4} & -L\sin\frac{\pi}{4} \\ -k & k & -k & k \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{15}$$

where k is the torque-thrust coefficient defined as $k = \frac{k_Q}{k_F}$, such that $Q_i = kT_i$. The motor control output is consequently obtained by inverting (15)

$$\Omega = \boldsymbol{B}^{-1}\boldsymbol{u} \tag{16}$$

Since the matrix $\boldsymbol{B}$ is constant, this inversion can be precomputed and implemented statically. Finally, the overall block diagram of the attitude controller can be seen in Fig. 8.

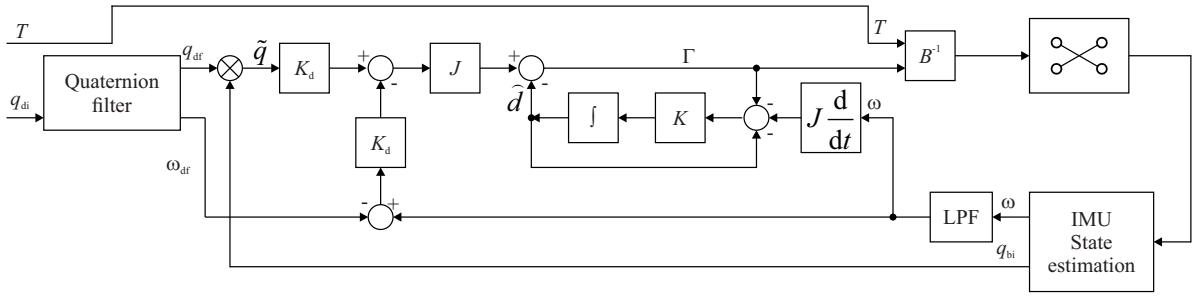

Fig. 8. Block diagram of the attitude controller and disturbance observer

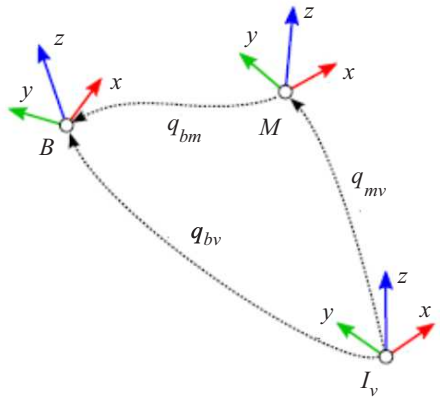

Fig. 9. Misalignment and transformation between marker frame M and body frame B

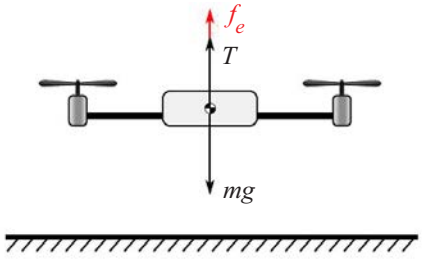

Fig. 10. Disturbance force during hover flight

3.3 Position control

The pose p_b of the quadrotor in the Vicon motion capture system frame I_V is described by the position $\boldsymbol{p}_b = [x\, y\, z]^\top$ and an orientation unit quaternion $\boldsymbol{q}_{\mathrm{bi}}$. The control input is obtained from the received desired pose commands, which consists of a desired position $\boldsymbol{p}_d = [x_d\ y_d\ z_d]^\top$ and a desired angular velocity around the yaw axis $\dot{\Psi}$. The goal of the position controller is to track the desired position $\boldsymbol{p}_d$ and a yaw angle Ψ around the inertial z -axis obtained by integrating $\dot{\Psi}$.

To accomplish this, a PD (proportional-derivative) controller with acceleration-based disturbance observer is implemented, which calculates a control force $\boldsymbol{f}$ in the inertial frame of reference. The update rate is chosen to be 100 Hz. Additionally, a third-order filter is applied to the position measurements $\boldsymbol{p}_b(t)$ in order to create a position, velocity and acceleration feedforward, which in turn improves the trajectory tracking performance. The pose of the quadrotor is tracked using LED-based and passive tracking markers, giving the orientation $\boldsymbol{q}_{mv}$ of the marker frame M in the frame I_V . Inevitably, a misalignment is present between M and the body frame B (Fig. 9). Therefore, to obtain the actual orientation $\boldsymbol{q}_{bv}$ of B in the frame I_V a calibration quaternion $\boldsymbol{q}_{bm}$ is introduced, so that $\boldsymbol{q}_{bv} = \boldsymbol{q}_{bm} \otimes \boldsymbol{q}_{mv}$. The virtual control force in the inertial frame of reference $\boldsymbol{f}_i = [f_x\ f_y\ f_z]^\top$ is calculated as

$$\boldsymbol{f}_i = m(-\boldsymbol{K}_d\boldsymbol{v} - \boldsymbol{K}_p(\boldsymbol{p}_b - \boldsymbol{p}_d)) + m\boldsymbol{g} \qquad (17)$$

where $\boldsymbol{K}_p, \boldsymbol{K}_d$ are the proportional and derivative diagonal gain matrices, respectively m is the nominal mass of the quadrotor and $\boldsymbol{g} = [0\ \ 0\ \ 9.81]^\top\,(\mathrm{ms}^{-2})$ is the gravitational acceleration vector.

A third-order filter with nominal frequency $\Lambda = 5\,\mathrm{rad/s}$ is implemented to realize the desired pose feedforward. It takes the received desired position $\boldsymbol{p}_{d,0}$ and produces the filtered desired position $\boldsymbol{p}_d$, desired velocity $\dot{\boldsymbol{p}}_d$ and the desired acceleration $\ddot{\boldsymbol{p}}_d$. Importantly, it is run for each axis independently. Therefore, the bold vector notation was omitted in the following filter definitions, since it is dealt with scalars only. The filter is defined as

$$\begin{bmatrix} \dot{p}_d \\ \ddot{p}_d \\ \dddot{p}_d \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\Lambda^3 & -3\Lambda^2 & -3\Lambda \end{bmatrix} \begin{bmatrix} p_d \\ \dot{p}_d \\ \ddot{p}_d \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Lambda^3 \end{bmatrix} p_{d,0}\,. \qquad (18)$$

The quadrotor velocity $\boldsymbol{v} = \dot{\boldsymbol{p}}_b$ was initially obtained by simply differentiating the position upon each pose packet arrival. However, this solution was very sensitive to variations in packet arrival timing, which were caused by the radio link re-transmission delays and by the fact that pose packets are not being sent to the quadrotor when the tracking target is occluded. To address these issues, an additional third-order filter is implemented, which takes the quadrotor body pose $\boldsymbol{p}_{b,0}$ and produces the filtered position $\widehat{\boldsymbol{p}}_b$, velocity $\widehat{\dot{\boldsymbol{p}}}_b$ and acceleration $\widehat{\ddot{\boldsymbol{p}}}_b$ signals. This filter is also run independently for each axis. It is defined analogically to (18) with the nominal filter frequency $\Lambda = 30\,\mathrm{rad/s}$.

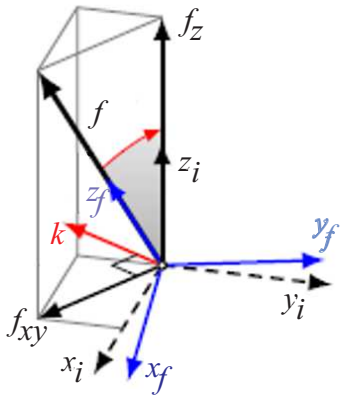

Fig. 11. Thrust transformation $\boldsymbol{q}_\mathrm{f}$

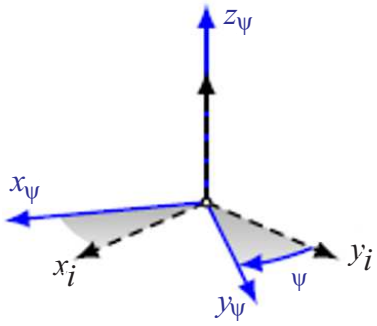

Fig. 12. Yaw transformation $\boldsymbol{q}_\Psi$ [18]

The control law (17) is afterwards modified to include the filtered quadrotor position and velocity and the feed-forward signals

$$\boldsymbol{f}_i = m(\ddot{\boldsymbol{p}}_d - K_d(\dot{\hat{\boldsymbol{p}}}_b - \dot{\boldsymbol{p}}_d) - \boldsymbol{K}_p(\hat{\boldsymbol{p}}_b - \boldsymbol{p}_d)) + m\boldsymbol{g}\,. \quad (19)$$

During flight, the gravitational force $\boldsymbol{f}_g = m\boldsymbol{g}$ is acting on the quadrotor body (Fig. 10). According to the Newton First law of motion, the velocity of the body to stay constant, the net force acting upon the body must be zero. Therefore, for the system model (1) it must hold that

$$m\ddot{\boldsymbol{p}} = m\boldsymbol{g}\boldsymbol{e}_3 + \boldsymbol{R}_{ib}\boldsymbol{f} + \boldsymbol{f}_e = 0\,. \quad (20)$$

This means the quadrotor must produce a thrust force $T = m\boldsymbol{g}$ in order to overcome the force of gravity and maintain a fixed position, assuming the ideal disturbance-free case (*ie* $\boldsymbol{f}_e = 0$). However, the quadrotor also experiences other external forces, such as aerodynamic drag and collisions. All such external influences can be represented by the external disturbance force $\boldsymbol{f}_e$. The most significant source of disturbance is the quadrotor weight m, for which the value of 0.033 kg is assumed. Since our quadrotors have different hardware configurations, their actual measured mass varies between approximately 0.029 kg and 0.039 kg. This mass difference is consequently perceived as a disturbance force $\widehat{\boldsymbol{f}}_e$. From the quadrotor dynamics model and assuming the condition (20), the position control disturbance force $\boldsymbol{f}_e^b$ in the body frame can be expressed as

$$\boldsymbol{f}_e^b = m\boldsymbol{a}^b - [\,0 \quad 0 \quad T\,]^\top \quad (22)$$

where $\boldsymbol{a}^b$ is the measured acceleration in the body frame including gravity and T (N) is the thrust force in the body frame. Since the control force f_i is already being calculated in the inertial frame, the disturbance force was chosen to calculate also in the inertial frame.

Therefore, (22) becomes

$$\boldsymbol{f}_e = m\boldsymbol{R}_{ib}\boldsymbol{a}^b - \boldsymbol{R}_{ib}\,[\,0 \quad 0 \quad T\,]^\top \quad (23)$$

where the rotation matrix $\boldsymbol{R}_{ib}$ describes the quadrotors current attitude in the inertial reference frame. For the implementation, rotation matrices were used instead of quaternions, however the concept could be applied analogically to quaternion-based attitude representation.

The disturbance force estimate $\widehat{\boldsymbol{f}}_e$ is afterwards obtained by low pass filtering the measured disturbance $\boldsymbol{f}_e$ in order to remove high frequency noise introduced by accelerometer measurements

$$\dot{\widehat{\boldsymbol{f}}}_e = \boldsymbol{K}_{DO,p}(\boldsymbol{f}_e - \widehat{\boldsymbol{f}}_e) \quad (24)$$

where $\boldsymbol{K}_{DO,p}$ is the gain of the low pass filter. Finally, the overall desired control force $\boldsymbol{f}$ is calculated as

$$\boldsymbol{f} = \boldsymbol{f}_i - \boldsymbol{f}_e\,. \quad (25)$$

To avoid excessive desired roll and pitch angles, the components f_x and f_y are limited to ± 0.4 N. The f_z component is limited to the range $\langle 0\,\mathrm{N}, 0.6\mathrm{N}\rangle$, which is the maximum thrust producible by the quadrotor [17]. This limits the desired angles to approx. $\pm 43°$, which allows highly aggressive flight maneuvers. The desired thrust T (N) is consequently obtained as $T = \|\boldsymbol{f}\|$. The desired attitude quaternion $\boldsymbol{q}_{di}$ can be obtained from the control force f using two transformations $\boldsymbol{q}_f$ and $\boldsymbol{q}_\Psi$, such as the desired attitude $\boldsymbol{q}_{di} = \boldsymbol{q}_f \otimes \boldsymbol{q}_\Psi$. The thrust transformation $\boldsymbol{q}_f$ aligns the z_b-axis to the control force $\boldsymbol{f}$ (Fig. 11). It is obtained by normalizing the quaternion $\boldsymbol{q}_f$, which is defined in [18] as

$$\boldsymbol{q}_f = \begin{bmatrix} f_z + \sqrt{1 + \boldsymbol{f}^\top \boldsymbol{f}} \\ -f_y \\ f_z \\ 0 \end{bmatrix}. \quad (26)$$

The yaw transformation $\boldsymbol{q}_\Psi$ rotates around the inertial axis z_i by angle Ψ (Fig. 12). It is obtained as [18]

$$\boldsymbol{q}_\Psi = \left[\,\cos\tfrac{\Psi}{2} \quad 0 \quad 0 \quad \sin\tfrac{\Psi}{2}\,\right]. \quad (27)$$

Figures 11 and 12 were used and modified with permission from [18].

3.4 Landing and collision detection

The concept of external force estimation can be further used to add interesting capabilities to the quadrotor. Such capabilities include the detection of landing, i.e. when the quadrotor is sitting on the ground or when is supported by an object from underneath. By high pass filtering the force estimation, events when the quadrotor impacts into (or is impacted by) another object while flying can be detected.

For landing detection, the disturbance force estimation $\widehat{\boldsymbol{f}}_e$ can be used directly. When the quadrotor is landed (*ie* sitting still on the ground), the motors are switched off and therefore the produced thrust force T is zero, meaning the only force acting on the quadrotor is the gravitational force $\boldsymbol{f}_g = m\boldsymbol{g}$. Again, to satisfy the condition from (22), it must hold that the disturbance $\boldsymbol{f}_e$ is equal to $\boldsymbol{f}_g$ in magnitude, but has the opposite direction. This also implies that the disturbance force is pointing upwards in the inertial frame.

The landing can be consequently detected by comparing the magnitude of the estimated disturbance force $\widehat{\boldsymbol{f}}_e$ to a defined threshold value and also by checking if the z-axis component of $\boldsymbol{f}_e$ is greater than zero. For the landing threshold a value of 0.11 N is used, which was determined experimentally such that up/down motion during normal flight is not detected as a landing. If these conditions are met, a LANDING_DETECTED event is triggered in the flight state machine.

Collision detection works on a similar principle. However, here the already estimated disturbance $\widehat{\boldsymbol{f}}_0$ can not be simply used, because its relatively slow low-pass filter would attenuate the short signal peaks caused by collisions. Therefore, the unfiltered measured disturbance fe was taken as it was introduced in (23). Then a separate, fast low-pass filter was applied, followed by a high-pass filter, which together effectively form a band-pass filter. The low pass filter is defined as

$$\widehat{\boldsymbol{f}}_{c,L} = \boldsymbol{K}_{C,L}\big(\boldsymbol{f}_e - \widehat{\boldsymbol{f}}_{c,L}\big) \tag{28}$$

where $\boldsymbol{K}_{C,L}$ is the filter gain. The high-pass filter is defined in discrete-time as

$$\widehat{\boldsymbol{f}}_{c,H}^{k+1} = \alpha\big(\boldsymbol{f}_(c,H)^k + \boldsymbol{f}_(c,L)^k - \boldsymbol{f}_(c,L)^{k-1}\big) \tag{29}$$

and α is the high-pass filter parameter defined by

$$\alpha = \frac{dt}{dt + t_H} \tag{30}$$

where t_H is the filter time constant and dt is the filter update period. Since the filter is updated within the position controller loop, its update rate is also 100 Hz. These two filters were tuned manually, such that they attenuate undesired high-frequency noise from the accelerometer and pass signals which correspond to a collision event. A collision can be detected by simply comparing the magnitude of the filtered disturbance $\widehat{\boldsymbol{f}}_{c,H}$ to a defined threshold, for which the value of 0.015 N was chosen. This threshold must be low enough in order to detect the relatively weak collision impulses, yet high enough so that regular oscillations of the quadrotor (caused by *eg* aerodynamic disturbance) are not detected. If the threshold is exceeded, at first the direction $\boldsymbol{c}_d$ of the incoming collision is calculated

$$\boldsymbol{c}_d = \frac{\widehat{\boldsymbol{f}}_{c,H}}{\|\widehat{\boldsymbol{f}}_{c,H}\|} \tag{31}$$

and a COLLISION_DETECTED event is triggered in the flight state machine, passing $\boldsymbol{c}_d$ as a parameter.

4 Results

During the tests, the quadrotors coordinate frame was kept aligned with the inertial frame I_V. This means that the quadrotors x-axis (roll) angle controls the movement in the translational y-axis and the quadrotors y-axis (pitch) angle controls the movement in the translational x-axis. The events where the plot line suddenly drops to zero are caused by momentary occlusions of the tracking targets. This happens naturally, especially when the tracking target is close to the ground, because afterwards the small tracking markers are at the maximum distance to the cameras, making it more difficult for the cameras to detect them. Moreover, given the placement of markers on the quadrotor frame, the propellers tend to occlude the markers while not spinning. Parameters of the attitude controller used during these tests are listed below. Controller gains are 3×3 diagonal matrices with the gains k_x, k_y, k_z on the diagonal, for which the abbreviated notation $\boldsymbol{K} = \mathrm{diag}\{k_x, k_y, k_z\}$ is used.

Parameters of the attitude controller were chosen as

$$\begin{aligned}
\boldsymbol{K}_p &= \mathrm{diag}\{64.0\,,\ 64.0\,,\ 64.0\} \\
\boldsymbol{K}_d &= \mathrm{diag}\{5.0\,,\ 5.0\,,\ 10.0\} \\
\boldsymbol{K}_{\mathrm{DO},a} &= \mathrm{diag}\{6.0\,,\ 6.0\,,\ 6.0\} \\
\alpha &= 36.0
\end{aligned}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are the proportional and derivative controller gains, respectively; $\boldsymbol{K}_{\mathrm{DO},a}$ is the attitude disturbance observer gain and α is the quaternion filter gain. Parameters of the position controller are

$$\begin{aligned}
\boldsymbol{K}_p &= \mathrm{diag}\{3.0\,,\ 3.0\,,\ 3.5\} \\
\boldsymbol{K}_d &= \mathrm{diag}\{4.5\,,\ 4.5\,,\ 6.0\} \\
\boldsymbol{K}_{\mathrm{DO},p} &= \mathrm{diag}\{2.5\,,\ 2.5\,,\ 3.0\} \\
m &= 0.033\,\mathrm{kg}
\end{aligned}$$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are the proportional and derivative controller gains, respectively; $\boldsymbol{K}_{\mathrm{DO},p}$ is the position disturbance observer gain and m is the nominal quadrotor mass. The parameters of attitude and position controller were obtained through experimentation and manual tuning. For the nominal mass m, a fixed value of 0.033 kg

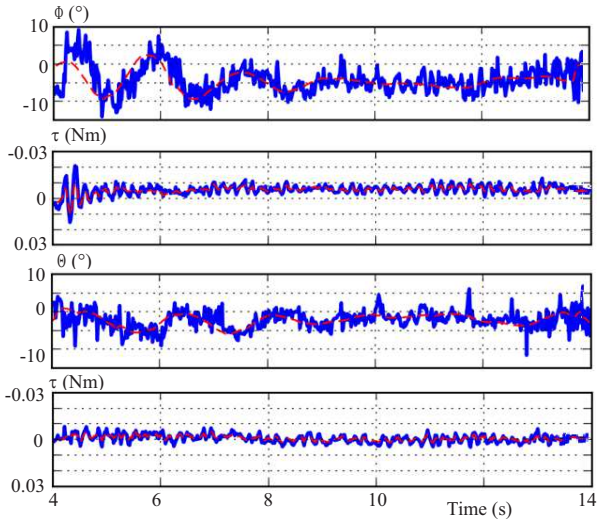

Fig. 13. Euler attitude, torque and disturbance during the hover flight

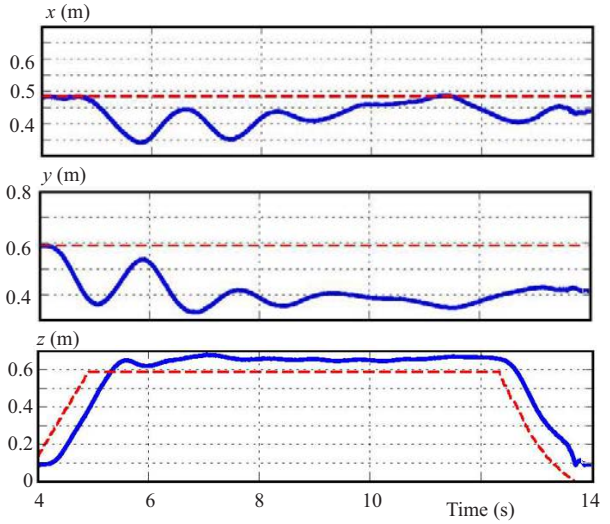

Fig. 14. Position tracking during the hover flight. RMSE = 0.129 m

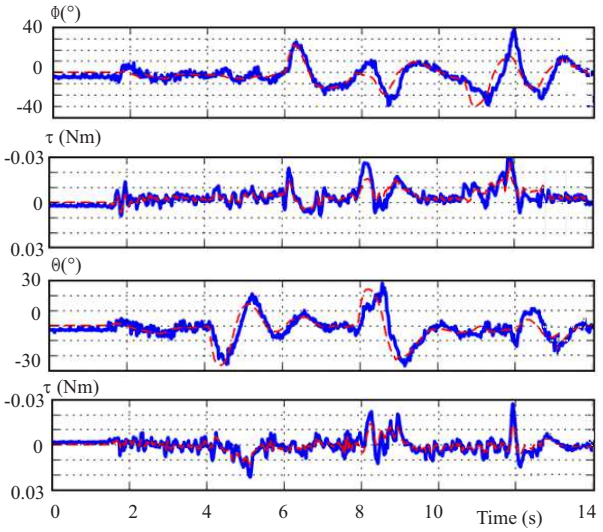

Fig. 15. Euler attitude, torque and disturbance during the step response flight

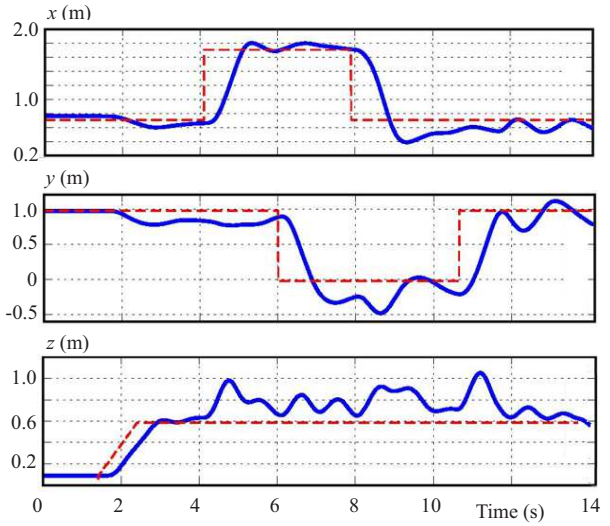

Fig. 16. Position tracking during the step response flight, RMSE = 0.263 m

for every quadrotor was assumed. However, each of our quadrotors has a different mass due to a different combination of flight battery, type of tracking markers (LEDs or passive) and not all quadrotors have the current measurement module attached to them. The mass of the quadrotor used in these tests is $m = 0.034$ kg. The position controller disturbance observer is capable to compensate this discrepancy and as a result the z height is being tracked with minimal error. A relatively large absolute offset is present in the x and y axis tracking, which was most likely caused by imprecise motion capture system calibration during these tests. This means that the motion capture system frame I_V is not perfectly aligned with the Earths gravity frame I_G and the quadrotor therefore receives incorrect orientation information. This introduces a certain attitude tracking error in the gravity frame, which exerts a force on the quadrotor and introduces an offset in the position tracking. Note however, that since the position control is calculated in the motion capture system frame, the disturbance observer can not detect this force and therefore can not compensate for it. This problem could be solved by adding an integral term into the position controller, improving the precision of the motion capture system calibration process, or by devising a method to calibrate the motion capture system frame I_V to Earths gravity frame I_G.

4.1 Hover flight

During the static hover flight test, the quadrotor is commanded to track the position $\boldsymbol{p_d} = [x_d\ y_d\ z_d]$ of the takeoff point. First, the takeoff command was issued, and the state machine increased the z_d height to cca 0.6 m. After the test, the land command was issued and the z_d height was brought down to $z_d = 0$ m by the state machine. Figure 13 shows tracking of the desired Euler angles Φ_d, Θ_d along with the desired control torques $\boldsymbol{\tau}_{\Phi,\Theta}$

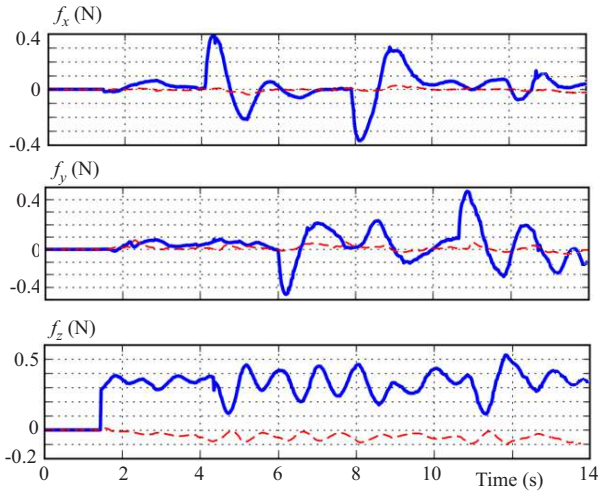

Fig. 17. Control force and disturbance during the step response flight

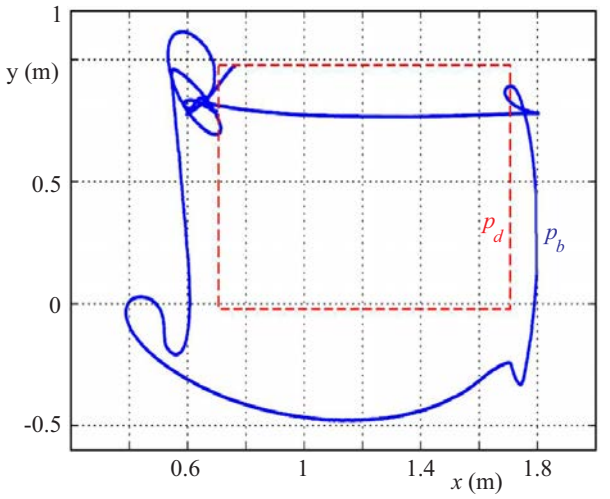

Fig. 18. Position tracking in the $x-y$ plane during the step response flight

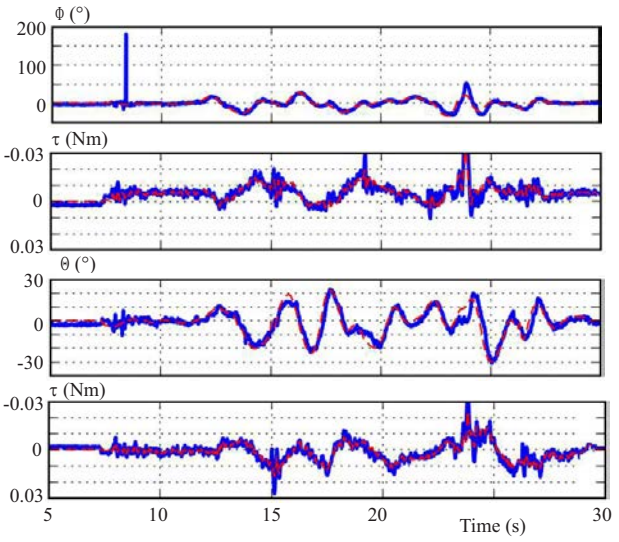

Fig. 19. Euler attitude, torque and disturbance during the manual control flight

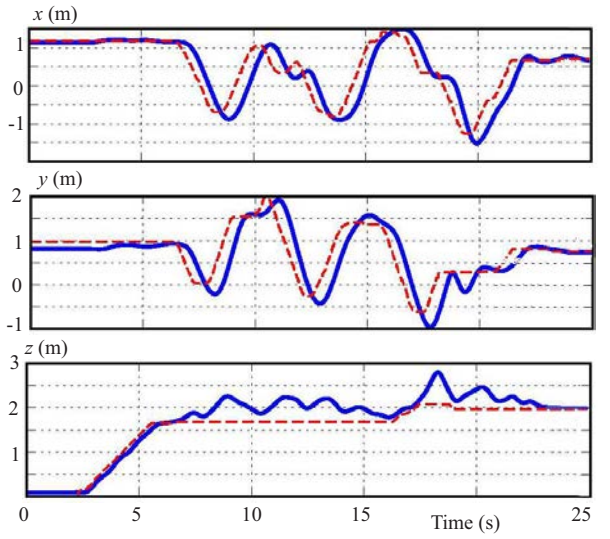

Fig. 20. Position tracking during the manual control flight, RMSE = 0.327 m

and the measured disturbances $\widehat{\boldsymbol{d}}_{\Phi,\Theta}$. Euler angles are calculated from the quadrotor pose $\boldsymbol{q}_{bi}$ and the desired attitude $\boldsymbol{q}_{di}$ generated by the position controller during the flight. Figure 14 shows tracking of the desired position $\boldsymbol{p}_d$. From this test it is concluded that the position controller is able to track the desired x, y position with a large absolute error up to 0.25 m due to the calibration problems described earlier, with relative error of less than ± 0.1m. The absolute error in the z-axis tracking is comparatively smaller than in the x and y axes thanks to the disturbance observer and the relative error stays within ± 0.1 m. Root mean square error (RMSE) of position tracking is 0.129 m.

4.2 Step response

During the step response flight test, takeoff is first commanded, the state machine increases the desired z-height to $z_d = 0.6$ m and the quadrotor tracks the initial position $\boldsymbol{p}_d$ at takeoff. Consequently, four step inputs with magnitude of 1 m are issued manually using the gamepad controller, forming a square trajectory. Figure 15 shows tracking of the desired Euler angles Φ_d, Θ_d along with the desired control torques $\boldsymbol{\tau}_{\Phi,\Theta}$ and the measured disturbances $\widehat{\boldsymbol{d}}_{\Phi,\Theta}$. Figure 16 shows tracking of the desired position $\boldsymbol{p}_d$. Figure 17 shows the calculated force $\boldsymbol{f}_i$ and the disturbance $\widehat{\boldsymbol{f}}_e$, and Figure 18 shows the position tracking in the $x-y$ plane. This test shows that the position controller is able to quickly reach the desired position, but significant overshoot and oscillation is present mainly in the y-axis. The position tracking offset mentioned at the beginning of this section is clearly visible in Fig. 18. Position tacking RMSE = 0.263 m was achieved. During the step transitions, the quadrotor achieves the maximum roll and pitch angles of $\pm 30°$ and momentarily peaking at $40°$, which is within the angle limits imposed by the force limits. The attitude controller is able to track the desired attitude generated by the position controller relatively precisely and without much overshoot. When

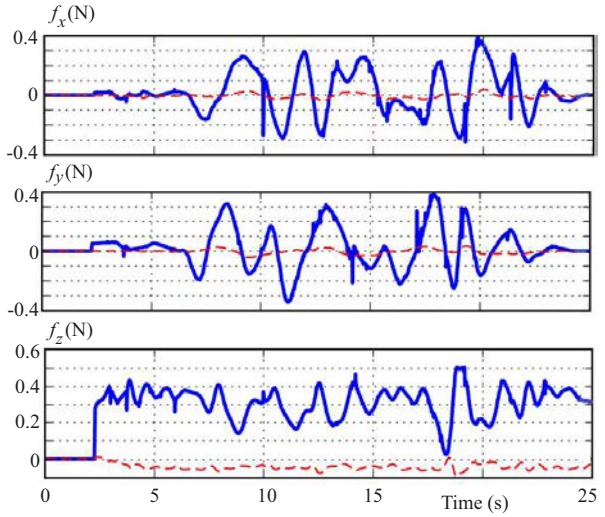

Fig. 21. Control force and disturbance during the manual control flight

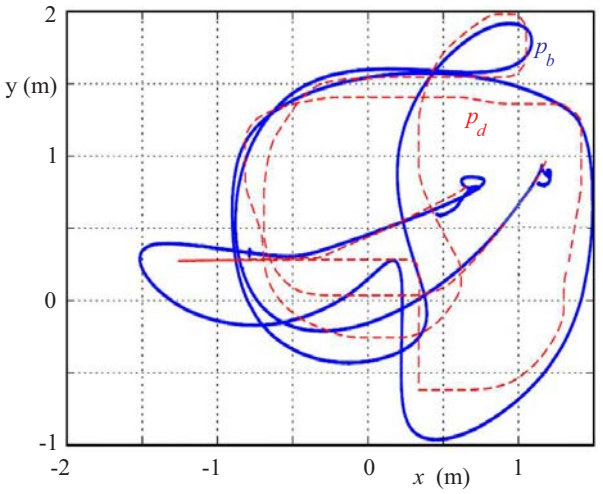

Fig. 22. Position tracking in the $x-y$ plane during the manual control flight

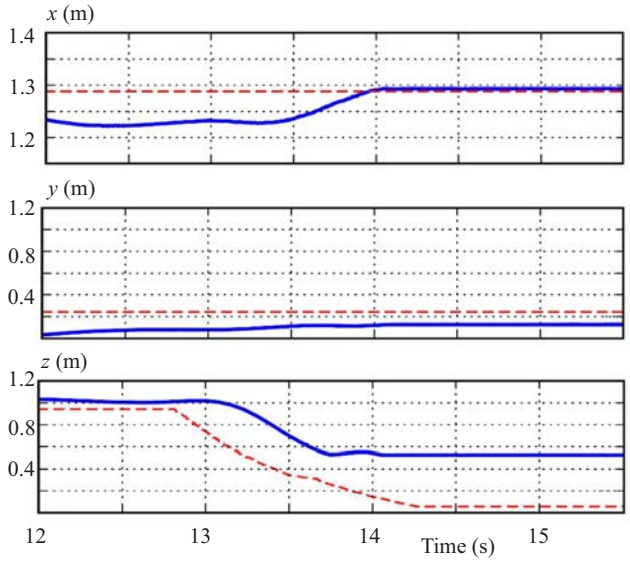

Fig. 23. Position during the landing detection test

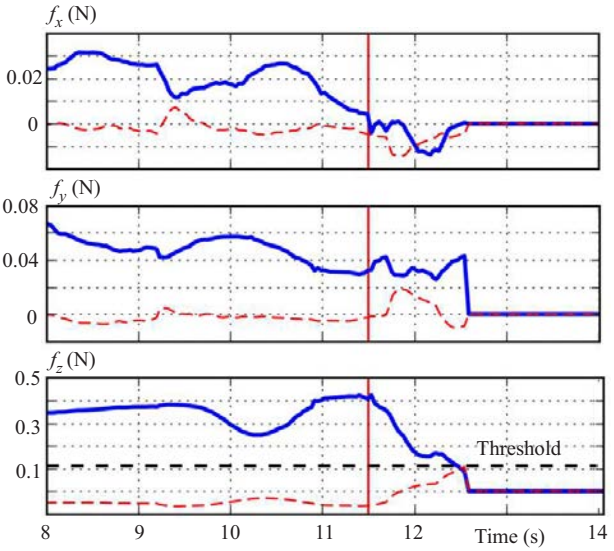

Fig. 24. Force and disturbance during the landing detection test.red vertical line marks the landing event at $t = 13.75$ s

a very sharp change in orientation is commanded by the position controller, such as at $t = 8$ s, attitude tracking is briefly worsened, because the motors reach saturation and the system therefore cannot generate the desired torque calculated by the attitude controller.

4.3 Manual control

The last test, which was performed, is flying under manual control. Here the quadrotor is commanded to take off and afterwards the desired position is controlled manually using the gamepad. The desired position setpoint is moving at maximum velocity of 1.5 m/s. Figure 19 shows tracking of the desired Euler angles Φ_d, Θ_d along with the desired control torques $\boldsymbol{\tau}_{\Phi,\Theta}$ and the measured disturbances $\widehat{\boldsymbol{d}}_{\Phi,\Theta}$. Figure 20 shows tracking of the desired position $\boldsymbol{p}_d$. Figure 21 shows the calculated force $\boldsymbol{f}_i$ and the disturbance $\widehat{\boldsymbol{f}}_e$, and Figure 22 shows the position tracking in the $x-y$ plane. This test serves to demonstrate, that the quadrotor is capable of fast flying, where aggressive changes in orientation are demanded. Compared to the previous tests, position tracking is improved. This is caused by the implemented position, velocity and acceleration feedforward control. In the previous tests, this generally introduces unwanted overshoot and oscillations, most likely because this controller and filter gains are not tuned perfectly. In this test however, the quadrotor is able to track the generated trajectory very precisely and with only minimal overshoot. Position tracking RMSE is higher at 0.327 m, due to the phase shift between the desired position and actual position signals.

4.4 Collision and landing detection

During the landing detection test, the quadrotor was flown over a roughly 0.5 m high obstacle and commanded to land. The state machine decreased the desired z-height at a certain velocity. As the quadrotor is descend-

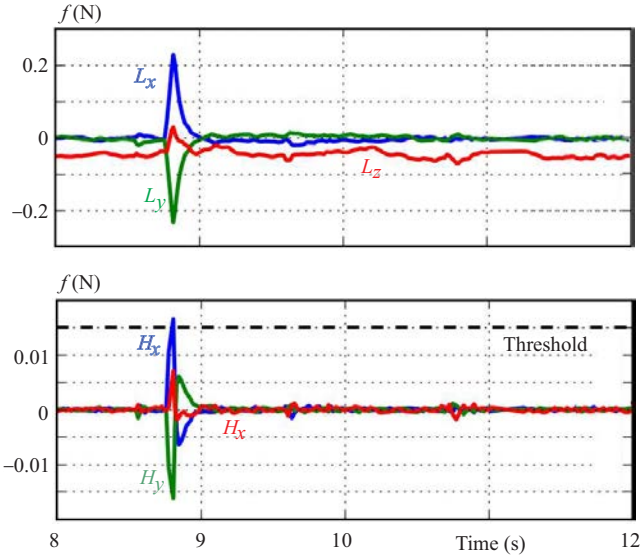

Fig. 25. Separate low-pass (top) and high-pass (bottom) disturbance filters for the collision detection

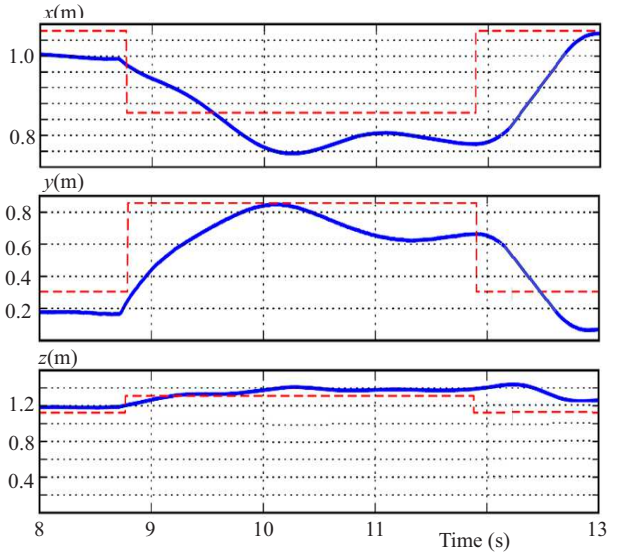

Fig. 26. Position during the collision detection test

ing (Fig. 23), the first contact occurs at $t = 13.75$ s (marked by the red vertical line), afterwards the quadrotor briefly bounces up and finally lands $t = 14$ s. Figure 24 shows the control force and disturbance during the test. Despite the physical landing occurred at $t = 14$ s, it takes a certain amount of time for the disturbance estimation $\widehat{\boldsymbol{f}}_{e,z}$ to reach the landing detection threshold, marked by the black dashed line. Afterwards the LANDING_DETECTED event is triggered in the flight state machine, which in turn disables the position controller and stops the motors.

During the collision detection test, the quadrotor is let to hover and track a certain desired position $\boldsymbol{p}_d$. Afterwards it was gently bumped by hand, by which a collision is simulated as it would happen during normal flight. The quadrotor is flying with its x-axis aligned with the inertial frames x-axis. Figure 25 shows the separate low-pass $\big(L(f_e)\big)$ and high-pass $\big(H(f_e)\big)$ filters used for the collision detection. The black dashed line marks the collision detection threshold.

Once it is reached, a COLLISION_DETECTED event is triggered in the flight state machine, passing the collision direction as a parameter. The state machine consequently adds a collision reaction offset to the desired position. This is clearly visible in Fig. 26. After a defined timeout (in this case 3 s), the state machine goes back to the normal flying state, the reaction offset is no longer applied and the quadrotor returns back to the original position $\boldsymbol{p}_d$.

5 Conclusion

The goal of this research was to develop a basic infrastructure for controlling a swarm of nano quadcopters with the ability to retrieve data from an on-board sensor. At first the platform for the experiments was selected - the Crazyflie quadcopter. A method to track the quadcopters in a motion capture system was devised. Overall systems architecture was proposed, described and evaluated. The Euler angle-based attitude controller was replaced by quaternion-based attitude controller with disturbance observer. Position controller with external disturbance force observer was developed, and the disturbance estimate was used to detect in-flight collisions and landing. Performance of the developed system was evaluated for hover flight, step response and manual control flight. Future work will mainly have to focus on solving the position control issues. To improve the overall flight performance, more effort could be put into better tuning of the controller and filter gains. However, the results allow the usage of a swarm of nano flying sensor nodes in numerous new applications. Results of this work can be also seen in a video at https://www.youtube.com/watch?v=LXng1v8lwbk.

Acknowledgments

This work and research has been carried out in the years 2015–2016 in German Aerospace Center - DLR, Robotics and Mechatronics Center (RMC). It was also supported by grants VEGA 1/0065/16, VEGA 1/0752/17 and APVV-14-0894.

References

[1] C. Lehnert and P. Corke, "μAV – Design and Implementation of an Open Source Micro Quadrotor", *Proceedings of the 2013 Australasian Conference on Robotics & Automation (ACRA2013)*, Australian Robotics & Automation Association, University of New South Wales, Sydney, pp. 1–8.

[2] S. Madgwick, "An Efficient Orientation Filter for Inertial and Inertial/Magnetic Sensor Arrays", *Report x-io and University of Bristol (UK)*, 25, pp. 113–118.

[3] Nordic Semiconductor, "Enhanced ShockBurst User Guide", available online: https://developer.nordicsemi.com/nRF5_SDK/nRF51_SDK_v5.x.x/doc/5.2.0/html/a00139.html (13th March 2014).

[4] Bitcraze, "Crazyflie Python Library", available online: https://wiki.bitcraze.io/doc:crazyflie:api:python: index (13th October 2015).

[5] Phenox Labs, "Phenox", available online: http://phenoxlab.com/?page_id=296.

[6] W. Hoenig, "radio_multilink", available online: https://github.com/whoenig/crazyflie-clients-python/tree/radio_multilink (3rd May 2015).

[7] T. Tomic and S. Haddadin, "Simultaneous Estimation of Aerodynamic and Contact Forces in Flying Robots: Applications to Metric Wind Estimation and Collision Detection", *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5290–5296.

[8] Vishay Semiconductors, "Vsmf4710", available online: http://www.vishay.com/docs/81470/ vsmf4710.pdf, Rev.1.4 (25^{th} September 2013).

[9] DeviationTX Project, available online: http://www.deviationtx.com/ (10^{th} June 2016).

[10] Vicon Motion Systems Ltd. Vicon, "Datastream SDK", available online: http://www.vicon.com/products/software/datastream-sdk.

[11] A. Kushleyev, V. Kumar and D. Mellinger, "Towards a Swarm of Agile Micro Quadrotors", *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[12] S. O. H. Madgwick, A. J. L. Harrison and R. Vaidyanathan, "Estimation of IMU and MARG Orientation using a Gradient Descent Algorithm", *Proceedings of the IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1-7.

[13] Maxim Integrated, "Max44284", available online: https://datasheets.maximintegrated.com/en/ds/MAX44284.pdf, Rev. 5 (November 2017).

[14] A. Collinson, "Infra Red Remote Control Extender", available online: http://www.zen22142.zen.co.uk/Circuits/Interface/irext.htm (24th May 2010).

[15] M. Furci, G. Casadei, R. Naldi, R. Sanfelice and L. Marconi, "An Open-Source Architecture for Control and Coordination of a Swarm of Micro-Quadrotors", *International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 139–146.

[16] S. Zhao, W. Dong and J. A. Farrell, "Quaternion-Based Trajectory Tracking Control of vtol-uavs using Command Filtered Backstepping", *American Control Conference*, June 2013, pp. 1018–1023.

[17] Bitcraze, "The Analysis of Finding a PWM to Thrust Transfer Function, available online: https://wiki.bitcraze.io/misc:investigations:thrust (15th July 2015).

[18] T. Tomic, "Evaluation of Acceleration-Based Disturbance Observation for Multicopter Control", *Control Conference (ECC)*, June 2014, pp. 2937–2944.

[19] A. Manecy, N. Marchand, F. Ruffier and S. Viollet, "X4-MaG: a Low-Cost Opensource Micro-Quadrotor and its Linux-Based Controller", *International Journal of Micro Air Vehicles*, 2015, vol. 7 no. 2, pp. 89–109.

[20] G. Loianno and V. Kumar, "Cooperative Transportation using Small Quadrotors using Monocular Vision and Inertial Sensing", *IEEE Robotics and Automation Letters*, vol. 3 no. 2, 2018, pp. 680–687.7.

[21] G. Pantelimon, K. Tepe, R. Carriveau and S. Ahmed, "Survey of Multi-Agent Communication Strategies for Information Exchange and Mission Control of Drone Deployments", *Journal of Intelligent & Robotic Systems*, 2018, pp. 1–10.

Received 1 November 2018