Networked control of a Parrot Mambo drone

Alexandru Codrean
Department of Automation
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
alexandru.codrean@aut.utcluj.ro

Attila Kovács
Department of Automation
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
attila.kovacs@aut.utcluj.ro

Octavian Stefan
Department of Automation and Applied Informatics
Politehnica University Timisoara
Timisoara, Romania
octavian.stefan@aut.upt.ro

Zsófia Lendek
Department of Automation
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
zsofia.lendek@aut.utcluj.ro

***Abstract*—The current study proposes a network control structure for small low-cost drones like the Parrot Mambo mini-drone. The structure is composed of an inner loop running on the drone, and an outer loop running on a remote computer. The inner loop controls the attitude and altitude of the drone based on Kalman filter estimations from the onboard sensors. The outer loop ensures position tracking based on measurements from OptiTrack cameras. A time delay compensator is added to address the constraints imposed by wireless network communications between the drone and the remote computer. Experimental results using Parrot Mambo drones show good stability and tracking performances, despite model uncertainty and time delay.**

***Index Terms*—Networked control, Parrot Mambo drone, Control applications**

I. Introduction

Small unmanned aerial vehicles (sUAVs) have become increasingly popular in multiple areas of applications, ranging from education to industry, from research and military to commercial [1]. The quadcopter (quadrotor) is the most common type of UAV used by inexperienced users, teachers, engineers and scientists, because it is inexpensive, lightweight, has easy-to-understand flight principles/high maneuverability, and a simple mechanical design [2]. Nevertheless, a quadcopter is an underactuated system (4 actuators and 6 degrees of freedom), nonlinear and inherently unstable, so designing a control law for stabilization and tracking is not a trivial task [3]. Moreover, real life applications impose complex challenges like state and input constraints, sensor noise and bias, reduced autonomy, and are easily affected by external disturbances. Furthermore, for small low cost drones there are additional hardware limitations, such as low quality sensors, reduced memory and computational capabilities, and network communication delays and packet loss. Although numerous control methods for drones have been investigated in the literature in the last two decades ([1], [2], [3]), many have been validated in simulations only. Other control approaches envisioning high performance - like model predictive control or nonlinear control [4] - use expensive and medium-to-large drones, capable of intensive computation onboard.

This work was supported by project DECIDE, no. 57/14.11.2022 funded under the PNRR I8 scheme by the Romanian Ministry of Research, Innovation, and Digitisation.

In this context the current research focuses on networked control for small low-cost drones with limited computational and communication capabilities, along with limited and unreliable sensors. To address these limitations, our approach splits the control structure in an inner loop and an outer loop: the inner loop is faster and runs on the drone, while the outer loop is slower and runs on a remote computer. The inner control loop is responsible for attitude and altitude stabilization based on estimates provided by a steady state Kalman filter, because the altitude and attitude can usually be reliably estimated based on measurements from a gyroscope and ultrasound sensors. Since obtaining accurate Cartesian position measurements is usually an issue with small drones without GPS capabilities, the outer control loop is closed over a wireless network by a remote computer, which is connected to motion capture cameras. This also solves the computation restrictions for the position tracking, and opens the way to further extensions involving trajectory planning and coordination of multiple drones. In order to cope with the issues caused by network communications [5], we consider a buffering technique that enforces a constant time delay, along with a time delay compensation method.

Our contribution lies in this new network control structure for small drones, with the outer loop closed through the network, along with the integration of a compensation strategy for network induced disturbances. The experiments done with the Parrot Mambo mini-drone show that our proposed approach is efficient in dealing with model uncertainty, unreliable sensing, as well as computation and communication constraints.

The structure of the paper is as follows: Section II describes the drone used and its mathematical model. Section III presents the networked control structure and states the control problem. Section IV designs the state estimator. Section V details the controller design, together with the time delay compensation. Section VI illustrates the experimental results.

Finally, Section VII draws some conclusions and mentions future research directions.

II. Preliminaries

Although many small low-cost drones are available commercially, in most cases the software onboard is not open-access, so there is no direct access to sensors and actuators, and the control and estimation algorithms can not be modified. A notable exception is the Parrot Mambo drone - see Figure 1, which has often been used in education and research [6]. For this reason we will use it as case study for the current paper. The following subsections will present the hardware and software features of the Parrot Mambo drone, along with its mathematical model.

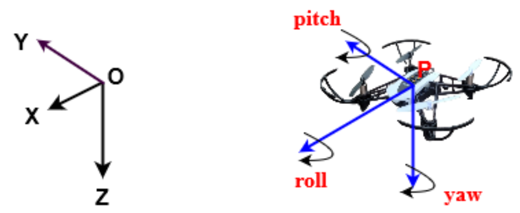


Fig. 1: Parrot Mambo Drone

A. Parrot Mambo drone

The Parrot Mambo drone is a lightweight small quadcopter of 0.18 $\times$ 0.18 meters, and weights only 0.063 kg. DC motors actuate the four propellers. The drone has the following onboard sensors: an accelerometer (measures the linear accelerations $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$), a gyroscope (Euler angle rates $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$), an ultrasound sensor (vertical distance from the ground z), a vertical camera, a barometer, and a temperature sensor. The vertical camera is used for estimating the velocities $\dot{x}$ and $\dot{y}$, using a combination of an optical flow and corner detection algorithms [7], and some geometric transformations and corrections [6]. The online communication with the drone is done wireless via Bluetooth, using the UDP transport protocol. The software interface with the sensors, actuators and network communication is made possible due to a firmware specifically developed for Matlab [8]. The drone has several limitations which were observed from our own experiments and reported also in the literature: i) the autonomy is very reduced, with a time of flight of maximum 2 minutes; ii) the limited memory and computational capabilities do not permit handling of large time varying matrices; iii) there are multiple issues with the optical flow measurements (see also [6] and [9]); iv) the Bluetooth communication has very limited bandwidth, and sending large packets at short time intervals (e.g. 5 ms) can cause delays of hundreds of milliseconds with packet loss exceeding 50%.

The control algorithm provided with the firmware [8] consists of multiple PID control loops, where angles and positions are estimated by linear steady state Kalman filters, all relying on simplified linear models. Due to optical flow measurements issues and yaw estimation issues, the drone can drift along the x and y axis, and has a constant yaw angle drift along the z axis.

B. Mathematical model of the drone

Let us consider the pose $P = [\xi \, \eta]^T = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T$ (see Figure 1), where the position in the world frame is given by $\{x, y, z\}$, and the orientation by the Euler angles ϕ (roll), θ (pitch), ψ (yaw). The dynamic model of the drone, developed using the Euler-Lagrange formalism, is given by [1]

$$\begin{cases} \ddot{x} = [c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi)]\dfrac{U_{coll}}{m} \\ \ddot{y} = [c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi)]\dfrac{U_{coll}}{m} \\ \ddot{z} = -g + c(\phi)c(\theta)\dfrac{U_{coll}}{m} \\ \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = J^{-1}(\eta)\left(\begin{bmatrix} U_\phi \\ U_\theta \\ U_\psi \end{bmatrix} - C(\eta, \dot{\eta})\dot{\eta}\right) \end{cases} \tag{1}$$

where $J(\eta)$ denotes the Jacobian matrix and $C(\eta, \dot{\eta})$ the Coriolis matrix. Due to space constraints, the full expressions are not given here, the interested reader is referred to [10].

The control inputs are the torques on the three rotational axes U_ϕ, U_θ, U_ψ, and the collective force U_{coll}. The nominal parameter values are: mass $m = 0.063$ kg, gravitational acceleration $g = 9.81$ m/s^2, X-axis inertia moment $I_x = 0.5829 \cdot 10^{-4}$ $kg\,m^2$, Y-axis inertia moment $I_y = 0.7169 \cdot 10^{-4}$ $kg\,m^2$, Z-axis inertia moment $I_z = 1.000 \cdot 10^{-4}$ $kg\,m^2$.

The nonlinear model (1) can be rewritten in state space form as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{2}$$

where the state vector is $\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ and the input vector is $\mathbf{u} = [U_{coll}, U_\phi, U_\theta, U_\psi]^T$. Let the equilibrium point in hovering mode be $\mathbf{x}^e = [x^e, y^e, z^e, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$, with inputs $\mathbf{u}^e = [U^e_{coll}, 0, 0, 0]^T$. The linearized model is

$$\begin{cases} \ddot{x} = \theta g \\ \ddot{y} = -\phi g \\ \ddot{z} = \dfrac{\Delta U_{coll}}{m} \end{cases} \qquad \begin{cases} \ddot{\phi} = \dfrac{U_\phi}{I_x} \\ \ddot{\theta} = \dfrac{U_\theta}{I_y} \\ \ddot{\psi} = \dfrac{U_\psi}{I_z} \end{cases} \tag{3}$$

where $\Delta U_{coll} = U_{coll} - m\,g$. The model can also be written as

$$\dot{\mathbf{x}}_l = A\mathbf{x}_l + B\mathbf{u}_l \tag{4}$$

where $\mathbf{x}_l = \mathbf{x} - \mathbf{x}^e$ and $\mathbf{u}_l = \mathbf{u} - \mathbf{u}^e$, and the expressions for the matrices A and B are omitted due to space restrictions.

In what follows we propose a networked control structure that can overcome some of the mentioned limitations of the Parrot Mambo drone.

[1] $s(\cdot)$ and $c(\cdot)$ are shorthand notations for $sin(\cdot)$ and $cos(\cdot)$.

III. Proposed networked control structure

The current paper proposes the networked control structure in Figure 2. An inner loop controller onboard the drone controls the attitude (Euler angles - pitch, roll, yaw) and altitude based on onboard estimates from a Kalman filter. An outer loop controller is implemented on a remote computer for XY position control, using the measurements from motion capture cameras, e.g. OptiTrack cameras. Because the command signals of the outer loop controller are sent through a wireless network to the drone, a time delay compensator is added to counteract the effect of network transmission delays. Note that the network is only on the direct path, because the position feedback is done thorough OptiTrack cameras, directly connected to the remote computer.

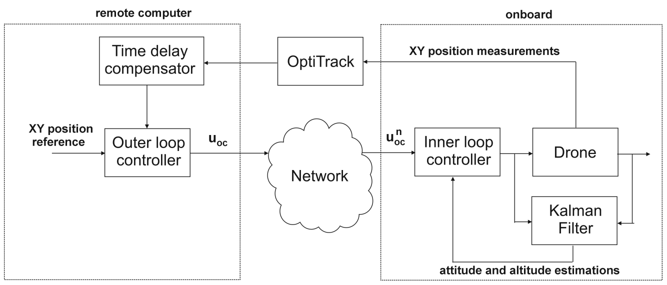


Fig. 2: Proposed networked control structure for the Parrot Mambo drone

The proposed structure has the following benefits:

- It bypasses the drone hardware limitations, by moving part of the control algorithm on a remote computer, as the outer control loop (which can run slower than the inner loop control).
- Accurate position measurements can be used from OptiTrack cameras, instead of onboard measurements and estimations through optical flow.
- The transmission delays and packet loss due to Bluetooth transmissions can be minimized by sending smaller data packets one way.

Splitting the control structure into an inner loop and outer loop relies on the assumption that the inner loop is much faster than the outer loop [2]. This can be ensured by properly tuning the inner/outer loop controllers and adopting the sampling periods. If for the inner loop control the sampling period is limited by the hardware to $T_s^i = 5\,ms$, for the outer loop the sampling period is limited by the Bluetooth transmissions and drone hardware. Our experiments revealed that a sampling period too small leads to a large amount of packet loss and large time-varying transmission delays (of several hundreds milliseconds, see also [11]). On the other hand, a large sampling period makes the control miss important transient behavior of the real drone, with control performance deteriorating significantly. After multiple experiments with the Parrot Mambo drone, we arrived empirically at a compromise solution for the outer loop sampling period of $T_s^o = 30ms$.

It is well known in the time delay systems literature that a relatively small time varying delay has often a more destabilizing effect on the control loop than a relatively large constant delay [12]. Consequently, in order to avoid time delay variations and packet loss due to Bluetooth transmissions, we adopt a queuing/buffering technique similar to the one in [13] in order to force a constant delay $\tau = n_\tau T_s^o$. The parameter n_τ depends on the amount of data the drone has to receive/transmit in a specific application.

Considering the network control structure from Figure 2, the Parrot Mambo drone described in Section II, and the network transmission aspects mentioned above, we can formulate the following two control problems (CPs):

CP_1. Design a local state estimator and a feedback controller for drone stabilization and attitude and altitude tracking, simple enough to run onboard.

CP_2. Design a remote XY position controller for tracking using OptiTrack camera measurements, together with a time delay compensator to counteract the effects of network transmissions effect on the control loop.

These will be addressed in the next two sections.

IV. State estimation

For the purpose of state estimation, we consider the discretized version of (4) affected by noise

$$\begin{cases} \mathbf{x}_l(k) & = A_d\mathbf{x}_l(k-1) + B_d\mathbf{u}(k-1) + \mathbf{w}(k-1) \\ \mathbf{y}(k) & = C_d\mathbf{x}_l(k) + \mathbf{v}(k), \end{cases} \tag{5}$$

where k denotes the current sample. The process and measurement noises, $\mathbf{w}$ and $\mathbf{v}$, are assumed to be independent white noises, with known covariances $E\{\mathbf{w}\mathbf{w}^T\} = \mathbf{Q}$ and $E\{\mathbf{v}\mathbf{v}^T\} = \mathbf{R}$.

The most commonly used state estimator for linear time invariant systems affected by noise is the linear Kalman filter [14], which we will employ in what follows. Our filter is different from the one already implemented through the firmware [8] because it uses the information provided by the control inputs. The filter equations can be found in [15], and are omitted here for reasons of brevity.

In order to exploit all available information, linear accelerations measured by the accelerometer are added as extra states. In this configuration, there are still three unobservable states - positions x, y and yaw (ψ) - which are calculated using numerical integration. During flight, the dissipated heat produces a bias in gyroscope measurements of pitch, roll and yaw rates [16]. This results in large estimation errors, especially for the roll and yaw angles. For the roll estimation, our solution is to add an additional state variable b to be estimated for the bias (a similar approach was adopted in [17]). The yaw rate bias was compensated by adding to the measured signal a correction term proportional with temperature. To summarize, the onboard Kalman filter estimates the states $[z, \phi, \theta, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}, \ddot{x}, \ddot{y}, \ddot{z}, b]^T$.

Due to the drone's hardware restrictions, we implemented a steady state version of the Kalman filter, with the steady state gain $\mathbf{K}_\infty$.

Since the ultrasound sensor can produce random false measurements during flight and take-off, for take-off, a Kalman filter that ignores measurements for the altitude z was used. Furthermore, during flight, measurement spikes that are above a given threshold are also eliminated, as they are likely false.

V. Control design and time delay compensation

In this section we describe the inner loop control, outer loop control and time delay compensation.

A. Inner loop control design

We design the inner control loop for attitude (Euler angles) and altitude (z position) control. Let the state vector for the inner loop control be $\mathbf{x}_i = [z, \phi, \theta, \psi, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$. Based on (3), the subsystem involving attitude and altitude can be written compactly as

$$\dot{\mathbf{x}}_i = A_i \mathbf{x}_i + B_i \mathbf{u}_l \tag{6}$$

A linear state feedback control is designed:

$$\mathbf{u}_l = -K_i \mathbf{e}_i = -K_i(\mathbf{x}_i - \begin{bmatrix} z_r \\ \phi_r \\ \theta_r \\ \psi_r \\ 0_{4\times 1} \end{bmatrix}), \tag{7}$$

where z_r is the altitude reference. For simplification, the desired altitude is considered constant. We also assume that the drone keeps the same orientation during position tracking, i,e. the yaw orientation reference ψ_r is taken as zero. ϕ_r and θ_r are regarded as virtual control inputs provided by the outer control loop through the network (see Figure 2): $\mathbf{u}_{oc}^n = [\phi_r \, \theta_r]^T$. The feedback gain K_i is designed using an LQR approach [18], with appropriate state and control inputs weights Q_i and R_i, see Section VI.

Remark 1: Although the altitude control is considered here as part of the inner loop, this does not limit the overall approach to tracking in the horizontal plane because the altitude reference z_r can still be sent remotely. Alternatively, the states z and $\dot{z}$ can easily be moved to the outer control loop without altering the overall framework.

B. Outer loop control design

For the outer loop control we consider that ideally $\phi \to \phi_r$, $\theta \to \theta_r$, $\psi \to \psi_r = 0$ and $z \to z_r = z^e$, and that the inner control loop is much faster than the outer loop control. Thus the outer loop can be modelled in a simplified manner, based on the first two equations from (3), as:

$$\dot{\mathbf{x}}_o = A_o \mathbf{x}_o + B_o \mathbf{u}_{oc} \tag{8}$$

with the state $\mathbf{x}_o = [x \, y \, \dot{x} \, \dot{y}]^T$ and input $\mathbf{u}_{oc} = [\phi_r \, \theta_r]^T$. At this point, we consider that there are no delays or packet loss, i.e. $\mathbf{u}_{oc} = \mathbf{u}_{oc}^n$. We add two extra states as the integrals of the tracking errors on the x and y axes

$$e_{ix} = \int_0^t (x_r - x)\, dt, \quad e_{iy} = \int_0^t (y_r - y)\, dt,$$

where x_r and y_r represent the reference positions. The extended system now becomes:

$$\dot{\mathbf{x}}_{eo} = A_{eo} \mathbf{x}_{eo} + B_{eo} \mathbf{u}_{oc} + B_r \mathbf{r} \tag{9}$$

with the matrices and vectors given by

$$\mathbf{x}_{eo} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ e_{ix} \\ e_{iy} \end{bmatrix}, A_{eo} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$B_{eo} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & g \\ -g & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, B_r = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{r} = \begin{bmatrix} x_r \\ y_r \end{bmatrix}.$$

For system (9), we consider the control law:

$$\mathbf{u}_{oc} = -K_{eo} \begin{bmatrix} \mathbf{x}_o - \delta \begin{bmatrix} \mathbf{r} \\ 0_{2\times 1} \end{bmatrix} \\ e_{ix} \\ e_{iy} \end{bmatrix}. \tag{10}$$

The feedback gain K_{eo} is again obtained through an LQR approach using an appropriate choice of the state and control inputs weights Q_{eo} and R_{eo} for $\mathbf{r} = 0_{2\times 1}$, see Section VI. The scalar $\delta > 0$ is introduced in order to account for the discrepancy between this ideal model and reality.

Remark 2: The control law (10) is equivalent with a combination of feedback and feedforward of the from $\mathbf{u}_{oc} = -K_{eo}\mathbf{x}_{eo} + \delta N \mathbf{r}$. The integrator component can bring the tracking errors $e_x = x_r - x$ and $e_y = y_r - y$ to zero, but at the price of a large overshoot due to nonlinearities and windup effect (command saturation), or a very large settling time. The role of the feedforward term in the control law is to improve the settling time, while also reducing the overshoot.

C. Time delay compensation

The outer loop is implemented remotely - see Figure 1, and as a result a network transmission delay appears in the loop. As discussed in Section III, the time delay can be made constant through a buffering strategy. One of the most effective time delay compensation strategy for constant and known delays is the Smith Predictor. Because our process (8) is unstable, we adopt a filtered Smith Predictor approach as in [19].

First we discretize (8) in series with a constant delay τ, through the zero order hold method [18], and obtain the following discrete transfer matrix:

$$\begin{bmatrix} X(z) \\ Y(z) \end{bmatrix} = \begin{bmatrix} H_x(z) & 0 \\ 0 & H_y(z) \end{bmatrix} \begin{bmatrix} \Theta_r(z) \\ \Phi_r(z) \end{bmatrix} \tag{11}$$

where

$$\frac{X(z)}{\Theta_r(z)} = H_x(z) = \frac{N_x(z)}{D_x(z)} z^{-d} = \frac{T_s^2 g}{2} \frac{z+1}{(z-1)^2} z^{-d},$$

$$\frac{Y(z)}{\Phi_r(z)} = H_y(z) = -\frac{T_s^2 g}{2} \frac{z+1}{(z-1)^2} z^{-d}, d = \frac{\tau}{T_s}.$$

Then, we rewrite the controller (10) as two discrete-time PID controllers with feedforward terms, having transfer functions $C_x(z) = \Theta_r(z)/E_x(z) + \delta N_x X_r(z)$ and $C_y(z) = \Phi_r(z)/E_y(z) + \delta N_y Y_r(z)$, where only the outputs X and Y are available.

Because the MIMO process (11) is decoupled and with the same delay, we design a filtered Smith predictor on each individual input-output channel. Consequently, we only show the design on the $\theta_r \to x$ channel (the design for the other channel can be done is similar manner).

Figure 3 presents the filtered Smith Predictor structure from [19]: 3a) represents the analysis structure, and 3b) represents the implementation structure. H_x^n is the nominal plant model and q_x is an input disturbance (due to external disturbances, uncertainty or unmodeled dynamics). The structure differs from the classical Smith predictor through the filter F_{rx}, which allows the process to have unstable poles, and increases the robustness of the overall structure to input disturbances.

Based on Figure 3a, the predicted signal when the input disturbance is zero can be determined as

$$\begin{aligned} X_p(z) &= H_x^n(z)\Theta_r(z) + F_{rx}(z)[X(z) - H_x^n(z)z^{-d}\Theta_r(z)] \\ &= F_{rx}(z)X(z) + [H_x^n(z) - F_{rx}(z)H_x^n(z)z^{-d}]\Theta_r(z) \\ &= F_{rx}(z)X(z) + S_x(z)\Theta_r(z). \end{aligned}$$

Since our process H_x has two unstable poles, and an additional step input disturbance due to uncertainty, a third order filter is adopted

$$F_{rx}(z) = \frac{N_{rx}(z)}{D_{rx}(z)} = \frac{a_3z^3 + a_2z^2 + a_1z + a_0}{(z-\alpha)^3}, \qquad (12)$$

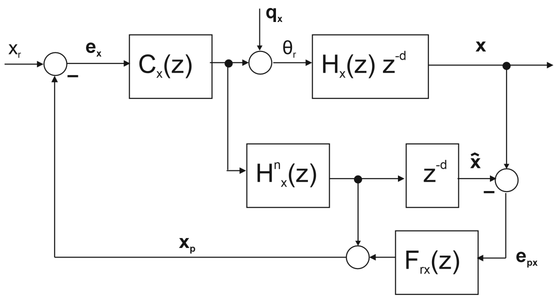
with $\alpha \in (0, 1)$ a tuning parameter. $N_{rx}(z)$ is designed such that the unstable poles of the process are eliminated from $S_x(z)$, and thus the implementation structure gives a stable prediction. In other words, we impose

$$\begin{aligned} 1 - F_{rx}(z)z^{-d} &= \frac{D_{rx}(z)z^d - N_{rx}(z)}{D_{rx}(z)z^d} \\ &= \frac{(z-z_0)(z-z_1)(z-z_2)p_x(z)}{D_{rx}(z)z^d}, \end{aligned}$$

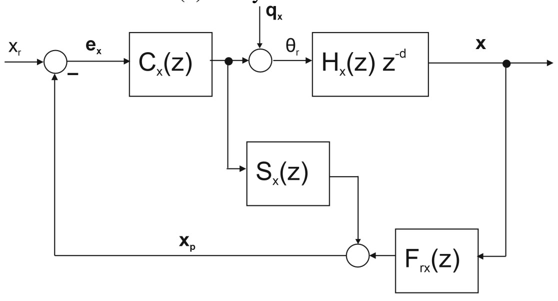
where $z_1 = z_2 = 1$ because of the double integrator process, and we additionally take $z_0 = 1$ so that $F_{rx}(1) = 1$ in steady state (step disturbance rejection). By dividing the polynomial $D_{rx}(z)z^d - N_{rx}(z)$ to $(z-1)^3$, we will get a quotient $p_x(z)$ and a remainder $r_x(z)$. The coefficients $\{a_3, a_2, a_1, a_0\}$ of N_{rx} are calculated by imposing the remainder to be zero.

VI. Experimental results

This section presents the actual design and the experimental results with the Parrot Mambo drone (along with OptiTrack cameras connected to a remote PC), using the structure from Figure 2, in the context of network transmission delays. As discussed in Section III, due to limitations of Bluetooth transmissions, depending on the amount data transmitted (e.g. images from the onboard camera), a certain delay $\tau = n_\tau T_s^o$ has to be taken into account. We consider here three scenarios: i) no delay case, i.e., $n_\tau = 0$, ii) a delay of 4 sampling periods, $n_\tau = 4$, and iii) a delay of 8 sampling periods, $n_\tau = 8$. The constant delay is enforced through the buffering technique explained in Section III.



(a) analysis structure



(b) implementation structure

Fig. 3: Filtered Smith predictor for time delay compensation on the $\theta_r \to x$ channel

First, the linear Kalman filter is designed. The covariance matrices $Q_k = diag(w_Q)$ and $R_k = diag(w_R)$ have been experimentally determined based on measured data, with $w_Q =$ [1e-4 1e-10 1e-10 1 0.1 5e-2 1e-5 1e-3 1 5e-4 5e-3 0.05 1e-11], $w_R =$ [0.7 300 300 5.6e-9 6.6e-8 1.3e-6 0.0147 0.0148 1.7e-5]. Second, the inner loop LQR was designed, using the weights $Q_i = diag([1\,0.5\,0.5\,0.001\,0.1\,0.04\,0.04\,0.01])$ and $R_i = diag([0.66\,10000\,10000\,1000])$. Third, the outer loop LQR was designed in the no delay case, with $Q_{eo} = diag([20\,20\,40\,40\,1800\,1800])$ and $R_{eo} = diag([5000\,5000])$, and using a feed-forward weight $\delta = 0.5$. Finally, the filtered Smith Predictor was designed for the case $n_\tau = 4$ (resulting in: $\alpha = 0.7, a_3 = 0, a_2 = 2.142, a_1 = -3.906, a_0 = 1.791$), and for $n_\tau = 8$ ($\alpha = 0.9, a_3 = 0, a_2 = 0.568, a_1 = -1.098, a_0 = 0.531$).

For all the experiments we considered the flight at a constant altitude, with a sequence of step reference signals on the x and y axes. So after the initial take-of with references ($x_r = 0, y_r = 0, z_r = 1m$), at time $t = 25sec$ a step reference of $-0.8m$ is applied on the x axis , then at $t = 30sec$ a step reference of $-0.5m$ on the y axis, and finally at $t = 35sec$ step signals on both x and y axes simultaneously ($x_r = -0.6m, y_r = -0.2m$).

Figures 4-6 present the experimental results from the moment of take-off. Reference signals are indicated by black lines, the measured output with delay compensation (our proposed approach) with blue and measured signals without delay compensation (for Figures 5-6) by dashed red lines.

Figure 4 illustrates the tracking results for the *no delay case* ($n_\tau = 0$). The initial oscillations are due to take-off, influence of unmodelled aerodynamics, OptiTrack cameras missing a marker, etc. The tracking results are satisfactory for a linear control structure on a low-cost drone, with no steady state error and small overshoot.

Figure 5 shows the tracking results for the *4 sampling periods delay case* ($n_\tau = 4$). The red-dashed trajectories are the measured outputs without the Filtered Smith Predictor, i.e. when there is no delay compensation. The sustained oscillations on long-term lead to crashes. The blue-solid trajectories are achieved using the Filtered Smith Predictor. It can be noticed that most of the original tracking performance (from the *no delay case*) is recovered.

Finally, the results for the *8 sampling periods delay case* ($n_\tau = 8$) are shown in Figure 6. Due to the large delay, the system becomes quickly unstable when there is no delay compensation, and the drone crashes or escapes the range of the OptiTrack cameras (red-dashed trajectories). Using the Filtered Smith Predictor for time delay compensations again recovers most of the original performances from the *no delay case* (blue-solid trajectories), despite small oscillations that emerge due to modelling and delay uncertainties. All in all, the experimental results show that the proposed networked control structure ensures good tracking performances despite model uncertainties, disturbances and relatively large delays (240 ms).

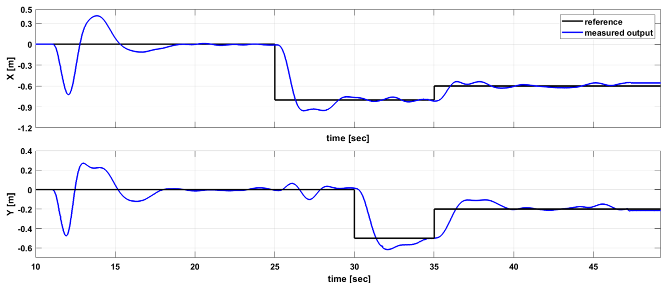


Fig. 4: Experimental results for the *no delay case*: tracking on the x and y axis at constant altitude of $1m$.

VII. Conclusions

The paper presented a networked control structure for low-cost drones like the Parrot Mambo, using OptiTrack cameras for accurate position measurements, and time delay compensation for addressing network communication constraints. A linear Kalman filter and an LQR controller is designed for the inner control loop running onboard the drone. The outer control loop is running on a remote computer, and consists of an LQR controller with integrator component and feedforward term, along with a Filtered Smith Predictor. The experimental results show that the control structure can ensure good tracking performances despite model uncertainties, disturbances, and relatively large delays. Our future work will focus on increasing the robustness of the control structure through unknown input observers, and resilience to cyber-attacks through AI inspired methods. Robustness is particularly important, since due to its small weight, the drone is vulnerable to persistent non-Gaussian disturbances, such as wind gusts, e.g., in outdoor scenarios.

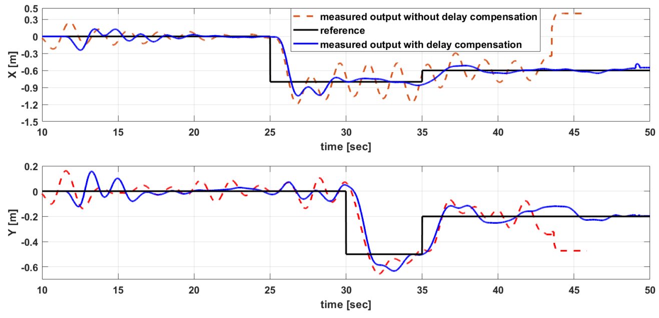


Fig. 5: Experimental results for a *delay of 4 samples periods* ($\tau = 120ms$): tracking on the x and y axis at constant altitude of $1m$.

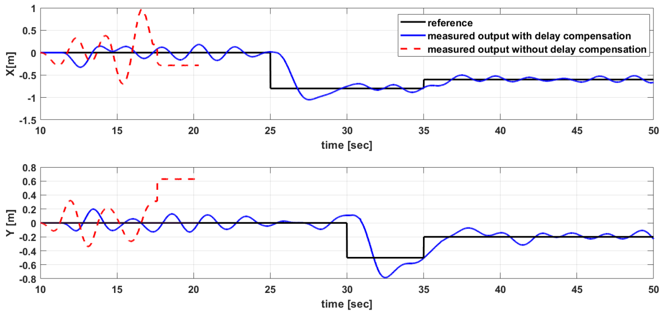


Fig. 6: Experimental results for a *delay of 8 samples periods* ($\tau = 240ms$): tracking on the x and y axis at constant altitude of $1m$.

References

[1] S. Mohsan, N. Othman, and Y. L. et al., "Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends," *Intel Serv Robotics*, vol. 16, no. 4, pp. 109–137, 2023.
[2] J. Marshall, W. Sun, and A. L'Afflitto, "A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems," *Annual Reviews in Control*, vol. 52, pp. 390–427, 2021.
[3] B. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control*, vol. 46, pp. 165–180, 2018.
[4] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, 2022.
[5] O. Stefan, T. Dragomir, A. Codrean, and I. Silea, "Issues of identifying, estimating and using delay times in telecontrol systems based on TCP/IP networks," *IFAC Proceedings*, vol. 43, no. 23, pp. 143–148, 2010.
[6] X. Zeng, *Implementing Tracking Error Control for Quadrotor UAV*. Master Thesis, Eindhoven University of Technology, 2021.
[7] T. Derbanne, *Method of evaluating the horizontal speed of a drone, in particular a drone capable of performing hovering flight under autopilot*. US Patent 8.498.447, 2013.
[8] Matlab, *Simulink Support Package for Parrot Minidrones*. Matlab, 2022.

[9] G. Brekelmans, *Extended Quadrotor Dynamics: from Simulations to Experiments*. Master Thesis, Eindhoven University of Technology, 2019.

[10] A.-K. Máthé, *Nonlinear Control for Commercial Drones in Autonomous Railway Maintenance*. PhD Thesis, Technical University of Cluj-Napoca, 2016.

[11] I. R. Scola, G. G. Reyes, L. G. Carrillo, J. P. Hespanha, and L. Burlion, “A robust control strategy with perturbation estimation for the Parrot Mambo platform,” *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1389–1404, 2021.

[12] O. Stefan, *Modelling, Analysis and Synthesis of Some Structures for Networked Control*. PhD Thesis, Politehnica University Timisoara, 2013.

[13] Y. Tipsuwan and M.-Y. Chow, “Control methodologies in networked control systems,” *Control Engineering Practice*, vol. 11, no. 10, pp. 1099–1111, 2003.

[14] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley & Sons, 2015.

[15] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*. TR 95-041, Department of Computer Science University of North Carolina at Chapel Hill, 2006.

[16] N. Metni, J.-M. Pflimlin, T. Hamel, and P. Soueres, “Attitude and gyro bias estimation for a flying UAV,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, AB, Canada, 2005, pp. 1114–1120.

[17] H.-J. Lee and S. Jung, “Gyro sensor drift compensation by Kalman filter to control a mobile inverted pendulum robot system,” in *2009 IEEE International Conference on Industrial Technology*, Churchill, VIC, Australia, 2009, pp. 1–6.

[18] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Pearson, 2020.

[19] R. C. Flesch, B. C. Torrico, J. E. Normey-Rico, and M. U. Cavalcante, “Unified approach for minimal output dead time compensation in MIMO processes,” *Journal of Process Control*, vol. 21, no. 7, pp. 1080–1091, 2011.