2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Event-Based Control for Embedded and Networked System
Application to a Mini Quadrotor Helicopter using Motion Capture

Sylvain Durand[*,†], Jonathan Dumon[†,*], Nicolas Marchand[†,*], J. Fermi Guerrero-Castellanos[‡]

[*] Univ. Grenoble Alpes, GIPSA-Lab, FR-38000 Grenoble, France.
[†] CNRS, GIPSA-Lab, FR-38000 Grenoble, France.
[‡] Autonomous University of Puebla (BUAP), Faculty of Electronics, MX-72570 Puebla, Mexico.

E-mail: sylvain@durandchamontin.fr

***Abstract*— Although periodicity simplifies design and analysis in control theory, it is no more adapted for embedded and networked cyber-physical systems because it results in a conservative usage of resources. Indeed, the control signal is computed and updated at the same rate regardless whether is really required or not, and is periodically sent on the communication link. On the other hand, event-driven sampling calls for resources whenever they are indeed necessary. An event-based controller is proposed in this paper as a solution to reduce the updates from the controller to the plant. An event-based corrector is also added to reduce the communications from the plant to the controller. The approach is tested for controlling the position of a real-time mini quadrotor helicopter using a motion capture system with deported controller. A reduction of the computing/communication resources utilization is highly demonstrated for similar final performance.**

INTRODUCTION

A *cyber-physical system* is an integration of computing devices with physical processes. In practice, embedded computers and networks monitor and control physical processes (usually with feedback loops) which, in return, affect computations and communications. The intersection between physical and information-driven (cyber) functions hence represents a challenge and results in innovation, see [12]. The use of *digital platforms* also emerges as an obvious trend to save space, weight and energy. However, their implementation can result in additional challenges, like determining how frequently the control signal needs to be updated and applied such that the stability properties are still guaranteed. Indeed, the consistently-used periodic fashion cannot be applied anymore in *embedded and networked systems* (with limited resources) and resource-aware implementations are required. In this context, recent works addressed alternative frameworks where the control law is event-driven. Whereas the control law is computed and updated at the same rate regardless whether is really required or not in the classical time-triggered approach, the *event-based paradigm* relaxes the periodicity of computations and communications in calling for resources whenever they are indeed necessary (for instance when the dynamics of the controlled system varies). Typical event-detection mechanisms are functions on the variation of the state (or at least the output) of the system, like in [1], [5], [19], [17], [2], [10], [14], [7], [4]. Although event-based control is well-motivated, only few works report theoretical results (about stability, convergence and performance) and practical implementation. It has notably been shown in [2] that the control law can be updated less frequently than with a periodic scheme while still ensuring the same performance. Stabilization of linear and nonlinear systems is analyzed in [22], [20], [15], [6], where the events are related to the variation of a Lyapunov function or the time derivative of a Lyapunov function (and consequently to the state too).

In the present paper, networked control systems (where the control loop is closed over a network) and their communication constraints are addressed. A deported controller has to control a cyber-physical system while reducing the communications between both (delays and packet losses are not considered here). Among many embedded and networked cyber-physical systems, Unmanned Aerial Vehicles (UAVs) have received growing interest in research. In particular, the *mini quadrotor helicopter* gives rise to great enthusiasm because of its high manoeuvrability, its payload capacity and its ability to hover [3]. The quadrotor is an under-actuated dynamic system with four input forces and six output coordinates (attitude and position). However, this system can be broken down into two subsystems, one defining the translation movement (position) and the other one the rotation movement (attitude). These subsystems are coupled in cascade since the translational subsystem depends on the rotational one, but the rotational subsystem is independent of the translational one. Nevertheless, event-based control is quite new for UAVs systems with high constraints, since the system has to be actively actuated to remain stable: attitude control was addressed in [21], [8], [9] and position control is addressed now. Furthermore, in order to release the platform from decision making related to guidance and navigation, position and orientation are calculated by a *motion capture* system here, where the movement of the vehicle is processed through high resolution cameras (based on the principle of inverse projection and triangulation).

The suggested setup to reduce the communications in such an architecture is divided into two parts. First, based on a seminal event-based PID controller (initially proposed in [1] and then improved in [5]), an *event-based RST controller* is proposed as a solution to reduce the communications from the controller to the plant. A RST digital controller [11], [16] can be seen as the discrete-time version of the well-known

978-1-4799-2376-2/14/$31.00 ©2014 IEEE

PID controller for first- and second-order controlled systems. However, it allows more tuning (because the RST controller allows independent specification of tracking and regulation performance whereas a PID only operates on the regulation error) and can be extended to systems of any order (this is not treated here). Then, an *event-based corrector* [13], [4] is also applied in order to reduce the communications from the plant to the controller. The idea is to make a copy of the plant model, on both sides of the network, and correct them when they deviate too much from the real system. The copy of the model in the controller side is used to compute the control law and the measurement is sent to the controller side over the communication link only when it has to be corrected. The rest of the document is organized as follows. In section I, preliminaries on PID and RST controllers are introduced. The event-based PID control developed in [5] is recalled and the new event-based RST strategy is detailed. The even-based corrector adapted from [4] to the RST scheme is also presented. The experimental platform is then depicted in section II. Experimental results highlight the capabilities of the proposed approach and a significant reduction of the communications. Discussions finally conclude the paper.

I. Event-based RST controller

A. From (time-triggered) PID to RST control

The continuous-time textbook PID controller is

$$u(t) = u_p(t) + u_i(t) + u_d(t) \quad (1)$$
$$\text{with} \quad \left| \begin{array}{l} u_p(t) = K_p e(t) \\ u_i(t) = K_i \displaystyle\int_0^t e(t)dt \\ u_d(t) = K_d \frac{de(t)}{dt} \end{array} \right.$$

where $u(t)$ is the control signal and

$$e(t) := y_{sp}(t) - y(t) \quad (2)$$

is the error between a given setpoint $y_{sp}(t)$ to track and the measurement of the controlled system $y(t)$. $u_p(t)$, $u_i(t)$ and $u_d(t)$ are the proportional, integral and derivative parts of the PID controller, where K_p, K_i and K_d are tunable parameters. A low-pass filter is also added in the derivative term (to avoid problems with high frequency measurement noise), where N is the filter gain hereafter. The control law is then written in the z-domain (more convenient for discrete-time systems), where the proportional part is straightforward, integral and derivative parts are discretized using the backward difference approximation (see [5] for more details). This gives

$$\left| \begin{array}{l} U_p(z) = K_p E(z) \\ U_i(z) = \dfrac{K_i \bar{h}}{1 - z^{-1}} E(z) \\ U_d(z) = \dfrac{K_d N (1 - z^{-1})}{(N\bar{h} + 1) - z^{-1}} E(z) \end{array} \right. \quad (3)$$

where $\bar{h}$ is the (constant) sampling period.

The RST digital controller form [11] is preferred in the sequel. Its architecture is depicted in Fig. 1. Actually, the

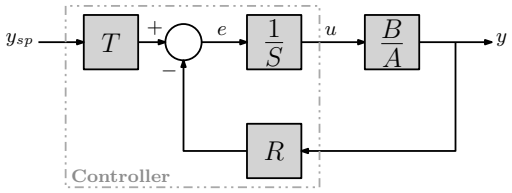


Fig. 1. RST controller setup.

RST controller allows independent specification of tracking and regulation performance (through T and R parameters respectively), whereas the PID controller only operates on the regulation error (which corresponds to the particular case $T = R$). Furthermore, whereas the PID design is restricted to first- and second-order systems, the RST can be extended to any order (but this is not treated here).

A period-dependent expression [16] is given here (since the sampling interval will vary in the sequel in the event-based scheme). In practice, R and S are obtained from the PID algorithm when developing (3) and considering $T = R$, leading to

$$\frac{R(z,\bar{h})}{S(z,\bar{h})} := \frac{r_0(\bar{h}) + r_1(\bar{h})z^{-1} + r_2(\bar{h})z^{-2}}{(1 - z^{-1})(1 - s_1(\bar{h})z^{-1})} \quad (4)$$

where

$$\left| \begin{array}{l} r_0(\bar{h}) = K_p + K_i\bar{h} + K_d N \delta(\bar{h}) \\ r_1(\bar{h}) = -K_p - \big[K_p + K_i\bar{h} + 2K_d N\big]\delta(\bar{h}) \\ r_2(\bar{h}) = \big[K_p + K_d N\big]\delta(\bar{h}) \\ s_1(\bar{h}) = \delta(\bar{h}) \end{array} \right. \quad (5)$$
$$\text{with} \quad \delta(\bar{h}) := \frac{1}{1 + N\bar{h}}$$

Consider a (exact or approximated) discrete-time second-order transfer function

$$G(z) = \frac{B(z)}{A(z)} := \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (6)$$

Note that the dependence on $\bar{h}$ for parameters a_1, a_2, b_1, b_2 is omitted here for the sake of simplicity, because the sampling period of the system to control will remain constant in the sequel (only the sampling interval of the control law will vary). Then, considering the system (6) and applying the RS control law (4), the closed-loop transfer function becomes

$$G_{cl}(z) = \frac{B(z)R(z,\bar{h})}{A(z)S(z,\bar{h}) + B(z)R(z,\bar{h})} \quad (7)$$

The RS control parameters r_0, r_1, r_2 and s_1 can be calculated directly in the discrete-time domain in such a way that $G_{cl}(z)$ matches with a desired closed-loop model. This can be done solving the *Bezout's equation* (also called *Diophantine equation*) for instance, see [11], [16]. However, in the present paper, the PID control parameters K_p, K_i, K_d and N are calculated in the continuous-time domain by pole placement, and so are then obtained the R and S parameters solving (5) for a given sampling period $\bar{h}$ (note that the same values for K_p, K_i, K_d and N are then kept when making varying the sampling interval in the sequel). Then, the zeros of the

closed-loop system (7) have to be considered to calculate the filtering transfer function T. The closed-loop transfer function with the complete RST control law becomes

$$G_{cl}(z) = \frac{B(z)T(z,\bar{h})}{A(z)S(z,\bar{h}) + B(z)R(z,\bar{h})} \tag{8}$$

Generally, no steady-state error is required, which yields

$$\frac{B(1)T(1,\bar{h})}{A(1)S(1,\bar{h}) + B(1)R(1,\bar{h})} = 1 \tag{9}$$

and since the controller (4) has an integrator, i.e. $S(1,\bar{h}) = 0$, then $T(1,\bar{h}) = R(1,\bar{h})$. Therefore, the simplest choice is

$$T(\bar{h}) = r_0(\bar{h}) + r_1(\bar{h}) + r_2(\bar{h}) \tag{10}$$

More complex solutions for T are not detailed here.

B. Event-based RST control

The approach is based on an original event-based PI controller, which setup was proposed for the first time in [1] and then improved in [5]. By *event-based PID control* we mean a set of two functions:

1) an event function ϵ, that indicates if one needs (when $\epsilon \leq 0$) or not (when $\epsilon > 0$) to recompute the control law;
2) a PID control law υ.

The event function is time-triggered with the sampling period $\bar{h}$ (that is the same as for the corresponding conventional time-triggered PID). In the present paper, an event is enforced when the *absolute error* crosses a given detection level $\bar{e}$ [5], this defines the event function as

$$\epsilon(t) = \bar{e} - \big|e(t)\big| \tag{11}$$

where $e(t)$ is defined in (2). On the other hand, the control signal is constant between two successive events

$$u(t) = \upsilon(t_k) \quad \forall t \in [t_k, t_{k+1}[\tag{12}$$

where t_k is a sampling instant (called an *event*) and, therefore, the length of the sampling intervals $h := t_k - t_{k-1}$ becomes not equidistant in time anymore.

Several event-based PI strategies are suggested in [5]. In particular, the *algorithm with exponential forgetting factor of the sampling interval* is applied here. The approach is somehow similar to the anti-windup mechanism used in control theory, where the error induced by the saturation has to be compensated. The integral part (in the z-domain) becomes

$$\begin{aligned} U_i(z) &= \frac{K_i\lambda^i(h)}{1-z^{-1}}E(z) \\ \text{with} \quad \lambda^i(h) &= he^{\alpha_i(\bar{h}-h)} \end{aligned} \tag{13}$$

where α_i is a degree of freedom to increase/decrease the exponential sampling interval of the integral part. One can refer to [5] for further details. Based on this seminal idea, an exponential forgetting factor is also applied in the derivative term, which yields

$$\begin{aligned} U_d(z) &= \frac{K_dN(1-z^{-1})}{\big[N\lambda^d(h)+1\big] - z^{-1}}E(z) \\ \text{with} \quad \lambda^d(h) &= \bar{h} + (h-\bar{h})e^{\alpha_d(\bar{h}-h)} \end{aligned} \tag{14}$$

where α_d is a degree of freedom to increase/decrease the exponential sampling interval of the derivative part. From these observations, an event-based RST controller is finally obtained

$$\begin{aligned} \frac{R(z,h)}{S(z,h)} &:= \frac{r_0(h) + r_1(h)z^{-1} + r_2(h)z^{-2}}{(1-z^{-1})(1-s_1(h)z^{-1})} \\ T(z,h) &= r_0(h) + r_1(h) + r_2(h) \end{aligned} \tag{15}$$

where

$$\left|\begin{array}{l} r_0(h) = K_p + K_i\lambda^i(h) + K_dN\delta\big(\lambda^d(h)\big) \\ r_1(h) = -K_p - \big[K_p + K_i\lambda^i(h) + 2K_dN\big]\delta\big(\lambda^d(h)\big) \\ r_2(h) = \big[K_p + K_dN\big]\delta\big(\lambda^d(h)\big) \\ s_1(h) = \delta\big(\lambda^d(h)\big) \end{array}\right. \tag{16}$$

where δ was defined in (5) for a constant sampling period, λ^i and λ^d are defined in (13) and (14) respectively.

C. Event-based corrector

Consider the system to control is such that

$$Y(z) = G(z)U(z) + H(z)P(z) \tag{17}$$

where Y and U are the (measured) output and (control) input of the system, P is an exogenous disturbance. The plant to control $G(z)$ is a discrete-time second-order transfer function as described in (6). The error on modeling and uncertainties are all lumped into the disturbance (not necessarily second-order) transfer function $H(z)$ which is not detailed here.

As already explained, a RST controller (4)-(5), (10) can make the closed-loop system (8) matches with a desired closed-loop model. However, in the present paper an event-based RST controller (15)-(16) is applied instead of the classical (time-triggered) controller. Furthermore, a networked control system is considered here and communications have to be reduced. For this reason, an event-based corrector [13], [4] is used to reduce the measurement transmissions (note that neither delays nor packet losses due to the communication link are considered here). The idea behind the event-based corrector is to have a copy of the system model (without disturbance) as defined in (6), on both sides of the network. The control law is calculated using the copy in the controller side, whereas the second copy is used in the plant side in order to detect when the model does not behave as the real system. Both copies are then updated with the real system output.

The system architecture with both the event-based controller and the event-based corrector is presented in Fig. 2.

1) Event generator for correction: This part runs a copy of the undisturbed system model (6). An event is then generated for correction when the difference between the (real) perturbed system output $y(t)$ and the output of the model copy $y_m(t)$ (in the plant node) reaches a given threshold $\bar{y}$, that is (in time domain) when

$$\big|y(t_j) - y_m(t_j^-)\big| = \bar{y} \tag{18}$$

where t_j^- is the time just before the event, and so is corrected the value of the event generator state such that

$$y_m(t_j^+) = y(t_j) \tag{19}$$

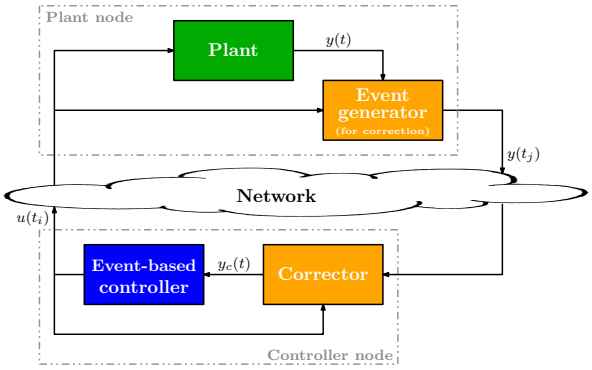


Fig. 2. System architecture.

where t_j^+ is the time just after the event. This defines the correction's event instant t_j. The system output $y(t_j)$ is then sent to the corrector (in order its model is also corrected) over the communication link.

2) Corrector: The corrector itself is in the controller node. It also runs a copy of the undisturbed system model (6) which has also to be updated when condition (18) is satisfied, which yields

$$y_c(t_j^+) = y(t_j) \tag{20}$$

where $y_c(t)$ is the output of the model copy in the controller node.

3) Event-based controller: In fact, the controller is not directly computed for the system to control without disturbance (6), neither for the model copy in the plant node, but for the copy of the model available in the controller node, that is the corrector model (6) with updates (20). The control's event instants t_i are hence determined by the vanishing of the event function (11) applied to y_c, which hence becomes

$$\begin{aligned} &\epsilon(t) = \bar{e} - \left|e_c(t)\right| \\ &\text{with} \quad e_c(t) := y_{sp}(t) - y_c(t) \end{aligned} \tag{21}$$

Also, the control law is no more computed using the error $e(t)$ but with $e_c(t)$ instead. The control signal $u(t_i)$ is then sent to the plant in order to be applied to both the plant and the event generator for correction.

II. Application to position control of a mini quadrotor helicopter using motion capture

A. Experimental platform

The algorithms are tested with a 18 grams Blade *Nano QX* quadricopter[1]. Its position and orientation are calculated by a *Vicon* motion capture system with T40s cameras[2] through Tracker software[3]. It is then sent to the control unit through a UDP frame every 2ms. Algorithms are programmed in *Matlab/Simulink* and implemented in real time at 200Hz to a target computer using *xPC target* toolbox. Finally, control variables are sent to the quadricopter through a GIPSA-lab's built-in bridge that converts UDP frames to DSMX 2.4Ghz protocol[4]. An overview of this architecture is presented in Fig. 3. The event-based RST controller is implemented in the control unit side. As regards the event-based corrector, whereas the event generator and the corrector should be implemented in the *Vicon* system side and the control unit side respectively, in practice both parts are also implemented in the control side in the present case. Indeed, it was decided to implement the event generator in the controller side for the sake of simplicity (because the *Vicon* unit algorithm is not accessible nor modifiable). As a consequence, all the blocks in Fig. 2 (except the plant) can be easily implemented in *Matlab/Simulink* in the control unit.

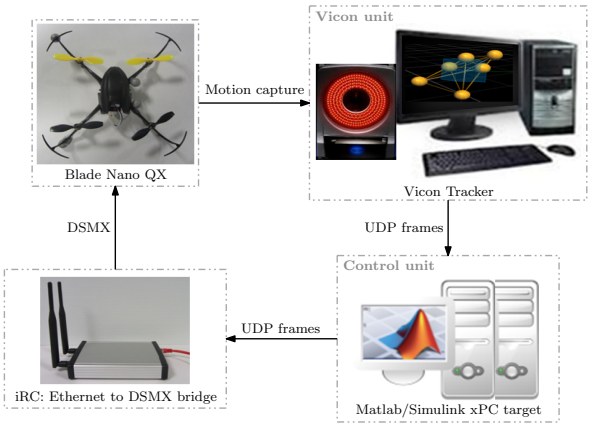


Fig. 3. Experiment architecture.

B. System model

The leveling control of the mini quadrotor helicopter (the control of pitch θ and roll ϕ angular velocities) is already done in a (non accessible) internal loop. Then, it is possible to control i) the yaw ψ angular velocity, ii) the altitude z and iii) the longitudinal V_{long} and lateral V_{lat} velocities. The aim here is to control the position x, y and z of the quadricopter (see Fig. 4 for a spacial representation). It is assumed that all the control variables can be independently controlled. It is also assumed that both longitudinal and lateral velocities similarly behave (they have the same model). The control of the yaw angle is not treated here.

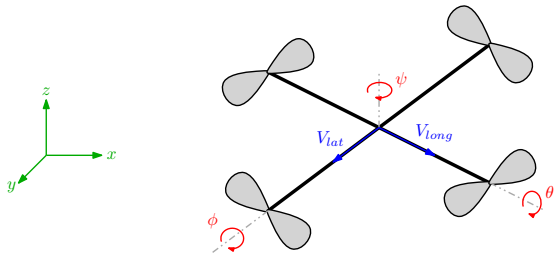


Fig. 4. Representation of the different control variables in the mini quadrotor helicopter.

[1]http://www.bladehelis.com/Products/Default.aspx?ProdID=BLH7680
[2]http://www.vicon.com/System/TSeries
[3]http://www.vicon.com/Software/Tracker
[4]https://www.spektrumrc.com/Technology/DSMX.aspx

Several experiments showed the system is highly nonlinear, with saturation and varying parameters (with respect to the battery power, the system aging, etc...), and is time delayed (delays are neglected here). Nevertheless, a simple model of the system has been obtained. The suggested model can be far from the real system but it will be shown in the sequel that the controller is robust enough and such an approximation allows to reduce the communications anyway. Actually, the system can be divided into several independent parts:

1) The longitudinal and lateral velocity, i.e. V_{long} and V_{lat} respectively, are (discrete-time) second-order transfer functions as defined in (6), that gives (after identification for the longitudinal case)

$$\frac{B(z)}{A(z)} := \frac{(4.691z^{-1} + 4.688z^{-2})10^{-7}}{1 - 1.998z^{-1} + 0.9981z^{-2}} \qquad (22)$$

for a sampling period $\bar{h} = 10\,ms$. A comparison between the modeled system response and real experiments for the same input signal is depicted in Fig. 5.
2) The altitude z is a double integrator system, which gain decreases with respect to the battery load (this is not detailed here).

One can remark that it has been decided in this paper to manage the position of the mini helicopter through a velocity control. For this reason, longitudinal and lateral velocity setpoints are dynamically build calculating the projection of the real position to the desired one, using information on x, y and ψ.

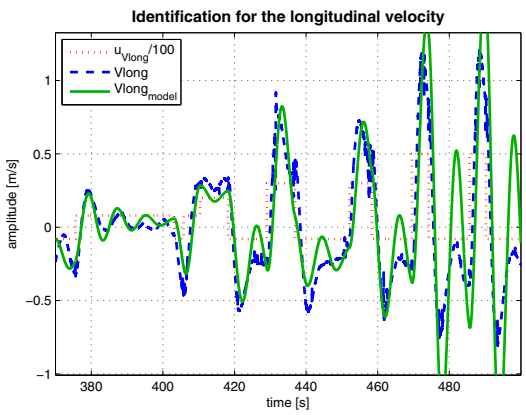


Fig. 5. Validation of the system model for the longitudinal velocity.

C. Performance indexes

Performance indexes, introduced in [18], are recalled here. They allow to compare the different event-based proposals with respect to classical approaches:

- The number (Nb) of samples required to perform the test bench.
- The IAE index, which gives information on the setpoint tracking:

$$IAE = \int_0^\infty \big|e(t)\big| dt$$

- The IAEP index, which compares the time-based and event-based system responses:

$$IAEP = \int_0^\infty \big|y_{tb}(t) - y_{eb}(t)\big| dt$$

where y_{tb} and y_{eb} are the time-based and event-based measurements respectively.
- The IAD index, which compares the time-based and event-based measurement errors:

$$IAD = \int_0^\infty \Big|\big|e_{tb}(t)\big| - \big|e_{eb}(t)\big|\Big| dt$$

where e_{tb} and e_{eb} are the time-based and event-based errors respectively.

The performance indexes obtained for the experiments detailed below are summarized in Table I. Results are discussed in the sequel.

D. Experimental results

The mini helicopter has to track three points in the space, which coordinates are $(x, y, z) = (0, 0, 0.6)$, $(1.2, 1.2, 0.6)$, and $(1.2, 0, 0.6)$. It was also decided that the system goes to another point after a waiting time of $40\,s$ in order to analyze both the point tracking and its stabilization near a given point. Note that the altitude is kept the same for the three points because the study focuses more on the longitudinal and lateral velocities control.

The system trajectory in the xy-plane are compared in Fig. 6 for several strategies:

a) a classical (time-triggered) PID controller;
b) an event-based PID controller (previously developed in [5]) where the control signal is calculated and updated only when the system output crosses a given level $\bar{e}$;
c) an event-based RST controller (proposed in the present paper, based on the event-based PID version);
d) an event-based RST controller using a copy of the system model (also proposed in the present paper) where the control law is computed using a copy of the system model, which is updated only when the error between the model and the real outputs crosses a given level $\bar{y}$.

The strategies are independently applied to control i) the altitude, ii) the longitudinal and iii) the lateral velocities. The control parameters K_p, K_i, K_d and N are calculated by pole placement (in the continuous-time domain) for the different controlled systems but they are identical in the different approaches. On the other hand, the event-based control parameters are $\bar{e} = 20\,mm$, $\alpha_i = 1$, $\alpha_d = 0.01$ for the altitude and $\bar{e} = 50\,mm/s$, $\alpha_i = 10$, $\alpha_d = 0.01$, $\bar{y} = 10\,mm$ for the velocities. Note that the copy of the model is not applied to the altitude but only to the control of velocities, using the identified model (22). The results in Fig. 6 are then discussed for each control variable z, V_{long} and V_{lat}.

1) Control of the altitude: Experimental results are represented in Fig. 7 for a $50\,s$ interval time. The top plot shows the setpoint and the measured signal whereas the bottom plot shows the sampling instants in the event-based control

TABLE I

PERFORMANCE INDEXES OBTAINED FOR THE DIFFERENT EXPERIMENTS WITH SEVERAL EVENT-BASED CONTROL STRATEGIES.

	z				Vlong				Vlat			
	Nb (%)	IAE	IAEP	IAD	Nb (%)	IAE	IAEP	IAD	Nb (%)	IAE	IAEP	IAD
Classical PID	100	0.55	0	0	100	2.74	0	0	100	1.34	0	0
Event-based PID	70.20	1.69	1.30	1.20	29.72	2.27	5.17	2.45	35.60	2.89	2.07	2.07
Event-based RST	46.66	1.07	0.83	0.70	45.06	2.75	3.11	2.70	44.36	3.21	4.73	2.32
Event-based RST with copies	–	–	–	–	39.60	2.63	4.17	2.08	58.10	4.70	5.56	3.60

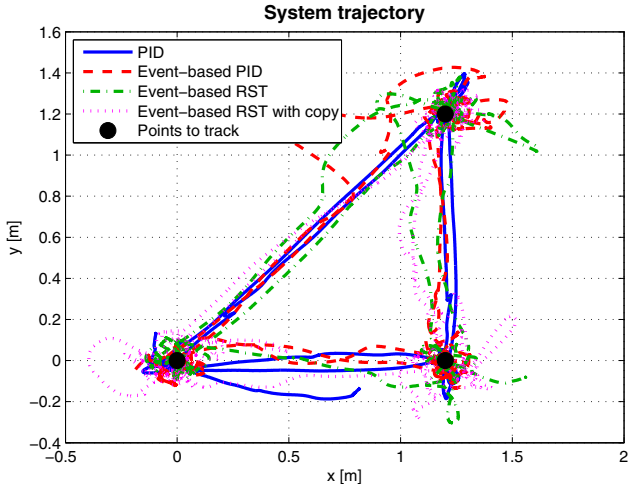
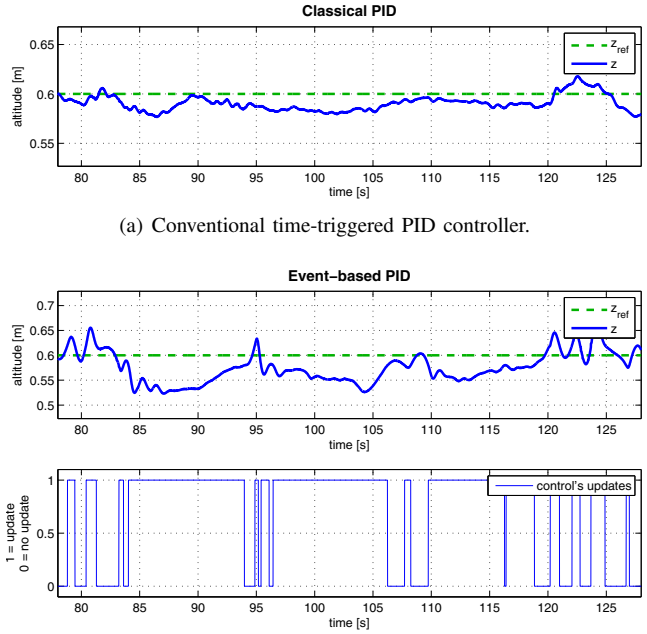
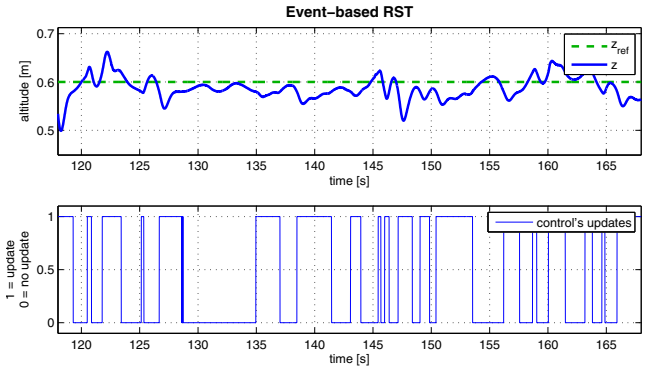


Fig. 6. Experimental results: comparison of the different strategies to track three points in the space.



(a) Conventional time-triggered PID controller.

(b) Event-based PID controller.



(c) Event-based RST controller.

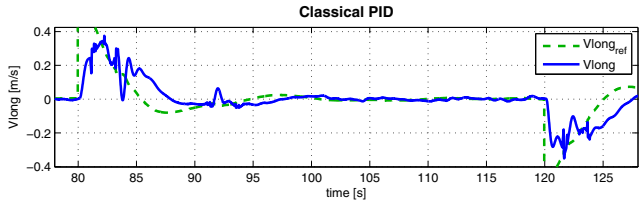
Fig. 7. Experimental results for the altitude.

schemes ('1' means the control law is calculated and updated during the sampling period $\bar{h}$, '0' means the control is kept constant).

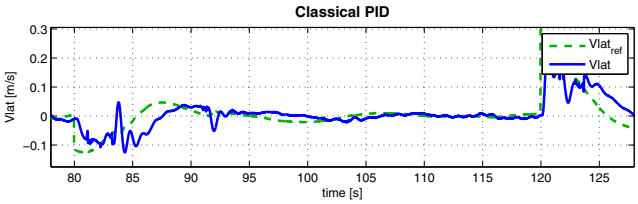
It can be seen that the altitude of the system is close to the desired $60\,cm$ altitude. The error is larger in the event-based schemes but with a strong reduction of the control updates in return (about $30\,\%$ and $55\,\%$ of samples less with the event-based PID and RST controller respectively than with a classical time-triggered strategy, see Table I). Furthermore, the event-based RST controller gives better results than the PID version (better IAE, IAEP and IAD indexes for a smaller number of updates) in the altitude case. Note that a better performance can be reached reducing the detection level $\bar{e}$.

2) Control of the longitudinal and lateral velocities: Remember the dynamical model is considered as the same for longitudinal and lateral velocities. Experimental results are represented in Fig. 8 and 9 respectively. As before, the top plot shows the setpoint and the measured signal whereas the bottom plot shows the sampling instants in the event-based control schemes. An extra plot in Fig. 8(d)-9(d) shows the correction instants in the event-based control scheme with copies of the system model ('1' means the measure is sent and the models are updated with the real measurement, '0' means the control is only computed using the model). Moreover, the output of the system model used to calculate the control law is also represented in the top plot.

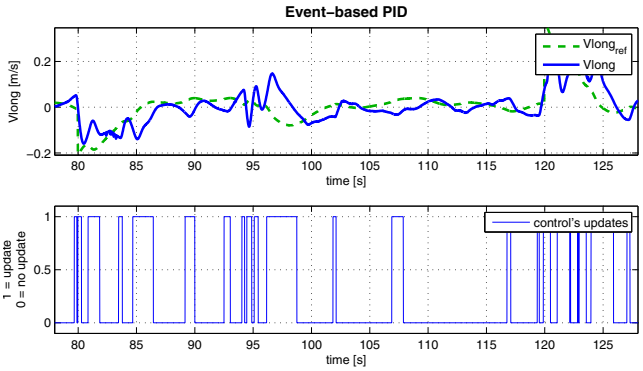
As regards the longitudinal velocity in Fig. 8, the frequency of control updates is also highly reduced with the event-based schemes (more than $55\,\%$). Note that since the setpoint to track is dynamically calculated, it varies from one experiment to another and it becomes difficult to compare the performance indexes. Nonetheless, it can be seen that IAE, IAEP and IAD indexes are close for the different event-based approaches (see Table I). As before, better performance can be obtained reducing the detection level $\bar{e}$. Furthermore, comparing the event-based RST controllers (without and with model copies), one can remark that the event-based corrector allows to reduce more i) the number of control updates ($12\,\%$ less) and ii) the correction updates ($70\,\%$ of



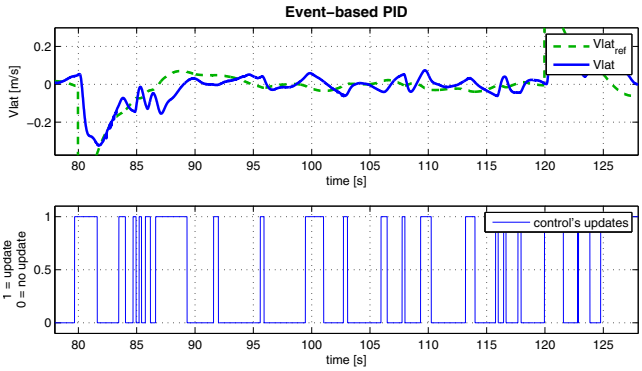
(a) Conventional time-triggered PID controller.



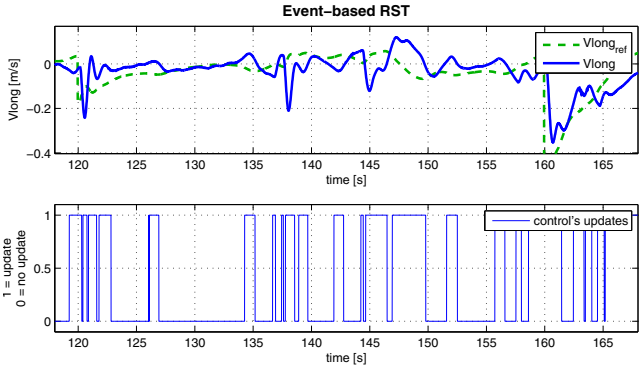
(a) Conventional time-triggered PID controller.



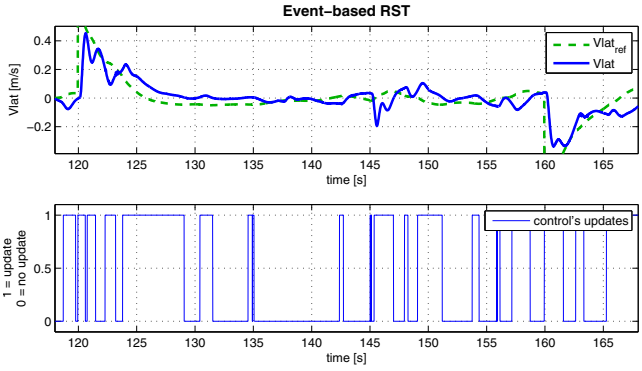
(b) Event-based PID controller.



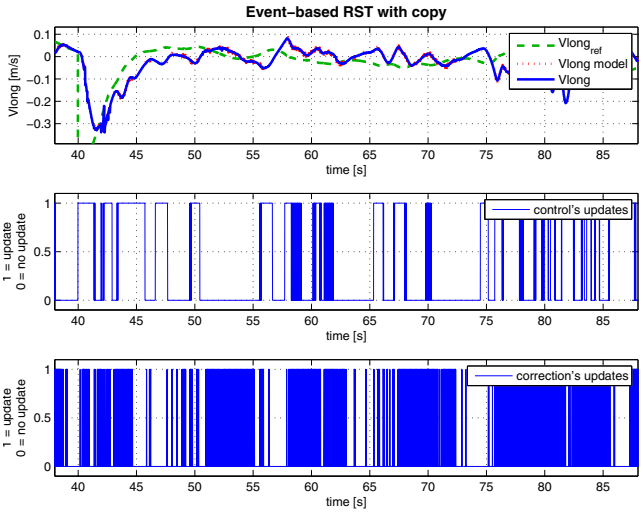
(b) Event-based PID controller.



(c) Event-based RST controller.

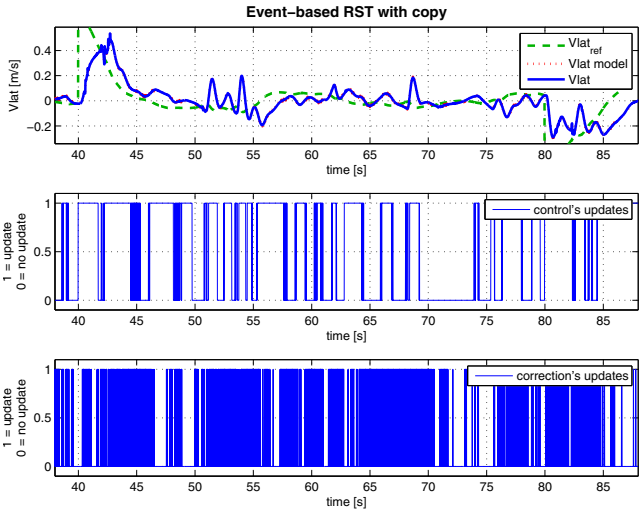


(c) Event-based RST controller.



(d) Event-based RST controller with copies.

Fig. 8. Experimental results for the longitudinal velocity.



(d) Event-based RST controller with copies.

Fig. 9. Experimental results for the lateral velocity.

communications less between the plant and the controller) even in the present case of a poor system model. The same remarks can be done for the lateral velocity (note that the performance are damaged because the model was computed for the longitudinal velocity and then simply applied to the lateral case considering that both are identical).

To sum up, the tradeoff between performance and the frequency of control/correction updates is clearly highlighted. One can hence imagine how the computing/communication resources utilization can be reduced in embedded and networked systems. This also means that the performance of the event-based schemes can be improved when decreasing the detection levels $\bar{e}$ and $\bar{y}$.

CONCLUSIONS AND FUTURE WORKS

This paper recalled the classical time-triggered PID control scheme as well as the event-based PID control paradigm presented in [5]. An event-based RST controller was then developed, where the control parameters are based on the PID scheme. An event-based corrector was also added. Both techniques allow to reduce the communications due to control, from the controller to the plant and from the plant to the controller respectively. The whole approach was tested on a real-time system: a mini quadrotor helicopter using a motion capture system to provide its position, where the controller is deported and communications are hence of high importance. Experimental results showed the effectiveness of the proposal with a high reduction of the computing/communication resources utilization. The advantage of an event-driven scheme was hence highlighted and the encouraging results strongly motivate to continue developing event-based control strategies.

Next step is to extend the proposed event-based RST controller to systems of any order (whereas only second-order systems were addressed here). Furthermore, the control parameters should be calculated directly in discrete-time domain (instead of continuous-time study). A better model of the system is also mandatory in order to implement more complex (event-based) strategies. Delays will also be considered in future works, as in [4]. Nonlinear strategies will also be a trajectory, with event-based control laws in the spirit of [15], [6]. Eventually, a cooperative approach where several systems are controlled together through a (wireless) network will be the next application to highlight the interest of event-based techniques and its strong reduction in the communications.

ACKNOWLEDGMENT

The authors would like to thank i) M. Fayet, D. Manica, P. Perraud and ii) T. Da Silva for their contribution in the context of an *integrative student project* and an *engineer internship* respectively (Ense3, Grenoble university). This work has been partially supported by the LabEx PERSYVAL-Lab (ANR–11-LABX-0025). The experimental platform has been designed by the GISPA-lab's technical staff and partially funded by EquipEx Robotex (ANR–10-EQPX-44-01).

REFERENCES

[1] K. E. Årzén. A simple event-based PID controller. In *Preprints of the 14th World Congress of IFAC*, 1999.

[2] K. J. Åström and B. Bernhardsson. Comparison of Riemann and Lebesque sampling for first order stochastic systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002.

[3] P. Castillo, A. Dzul, and R. Lozano. Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Transactions on Control Systems Technology*, 12:p. 510516, 2004.

[4] S. Durand. Event-based stabilization of linear system with communication delays in the measurements. Proceedings of the American Control Conference, 2013.

[5] S. Durand and N. Marchand. Further results on event-based PID controller. In *Proceedings of the European Control Conference*, 2009.

[6] S. Durand, N. Marchand, and J. F. Guerrero Castellanos. Event-based stabilization of nonlinear time-delay systems. In *Proceedings of the 19th World Congress of IFAC*, 2014.

[7] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos. Event-triggered control for discrete-time systems. In *Proceedings of the IEEE American Control Conference*, 2010.

[8] J. F. Guerrero-Castellanos, J. J. Téllez-Guzmán, S. Durand, N. Marchand, and J. Álvarez Muñoz. Event-triggered nonlinear control for attitude stabilization of a quadrotor. In *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems*, 2013.

[9] J. F. Guerrero-Castellanos, J. J. Téllez-Guzmán, S. Durand, N. Marchand, J. Álvarez Muñoz, and V. González-Díaz. Attitude stabilization of a quadrotor by means of event-triggered nonlinear control. *Journal of Intelligent and Robotic Systems*, pages 1–13, 2013.

[10] W. P. M. H. Heemels, J. H. Sandee, and P. P. J. van den Bosch. Analysis of event-driven controllers for linear systems. *International journal of control*, 81:571–590, 2009.

[11] I. Landau. The R-S-T digital controller design and applications. *Control Engineering Practice*, 6:155165, 1998.

[12] E. Lee and S. Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. Lee and Seshia, 2011.

[13] D. Lehmann and J. Lunze. Event-based control with communication delays. In *Proceedings of the 18th IFAC world congress*, 2011.

[14] J. Lunze and D. Lehmann. A state-feedback approach to event-based control. *Automatica*, 46:211–215, 2010.

[15] N. Marchand, S. Durand, and J. F. Guerrero-Castellanos. A general formula for event-based stabilization of nonlinear systems. *IEEE Transactions on Automatic Control*, 58(5):1332–1337, 2013.

[16] D. Robert, O. Sename, and D. Simon. Sampling period dependent RST controller used in control/scheduling co-design. In *Proceedings of the World Congress of IFAC*, 2005.

[17] J. Sánchez, M. Guarnes, and S. Dormido. On the application of different event-based sampling strategies to the control of a simple industrial process. *Sensors*, 9:6795–6818, 2009.

[18] J. Sánchez, M. Guarnes, S. Dormido, and A. Visioli. Comparative study of event-based control strategies: An experimental approach on a simple tank. In *Proceedings of the European Control Conference*, 2009.

[19] J. H. Sandee, W. P. M. H. Heemels, and P. P. J. van den Bosch. Event-driven control as an opportunity in the multidisciplinary development of embedded controllers. In *Proceedings of the American Control Conference*, pages 1776–1781, 2005.

[20] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52:1680–1685, 2007.

[21] J. J. Téllez-Guzmán, J. F. Guerrero-Castellanos, S. Durand, and N. Marchand. Event-based LQR control for attitude stabilization of a quadrotor. In *Proceedings of the 15th IFAC Latinamerican Control Conference*, 2012.

[22] M. Velasco, P. Martí, and E. Bini. On Lyapunov sampling for event-driven controllers. In *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009.