Event-Based PID Control: Application to a Mini Quadrotor Helicopter

Sylvain Durand, Bruno Boisseau, Nicolas Marchand, Jose-Fermi Guerrero-Castellanos

Event-Based PID Control:
Application to a Mini Quadrotor Helicopter

Sylvain Durand * **Bruno Boisseau** ** **Nicolas Marchand** **
J. Fermi Guerrero-Castellanos ***

* *ICube, INSA Strasbourg, Univ. Strasbourg, CNRS (UMR 7357), Strasbourg, France (e-mail: sylvain@durandchamontin.fr).*
** *GIPSA-lab, Univ. Grenoble Alpes, CNRS (UMR 5216), Grenoble, France.*
*** *Autonomous University of Puebla (BUAP), Faculty of Electronics, Puebla, Mexico.*

Abstract: Although periodicity simplifies design and analysis in control theory, it is no more adapted for embedded 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. On the other hand, event-driven sampling calls for resources whenever they are indeed necessary. Event-based PID controllers are proposed in this paper as an alternative to classical PID approaches, with the same performance but reducing the control updates. The algorithms are built here without safety limit condition contrary to the seminal event-based PID setup that was originally proposed by Årzén (1999), in order to reduce the periodicity even more. Both integral and derivative terms are considered. The different approaches are tested for controlling the position of a real-time mini quadrotor helicopter. A reduction of the computing and communication resources utilization is demonstrated for similar final performance.

Keywords: Event-based control, PID control, mini quadrotor helicopter.

1. INTRODUCTION

Advances in fabrication and design of VLSI (very-large-scale integration) circuits have resulted in the availability of low-cost, low-power, small-sized computational elements that are able to communicate via shared and possibly wireless communication network. Furthermore, development in MEMS (microelectromechanical systems), which provide solid-state sensors and actuators, complements these advances. The net result is ubiquitous sensing, computation, actuation and communication that allow the development of the so called *cyber-physical systems (CPS)*. These CPS represent 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 use of small-sized computational elements emerges as an obvious trend to save space, weight and energy. However, their implementation can result in additional challenges since the traditional feedback loop that operates in continuous-time or at a fixed sampling rate cannot be used anymore. Therefore, 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 Årzén (1999); Durand and Marchand (2009); Sandee et al. (2005); Sánchez et al. (2009a); Åström and Bernhardsson (2002); Heemels et al. (2009); Lunze and Lehmann (2010); Eqtami et al. (2010). Although event-based control is well-motivated and theoretical results have been stated in the literature in the last decades, only few works report practical implementation. It has notably been shown in Åström and Bernhardsson (2002) 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 Velasco et al. (2009); Tabuada (2007); Marchand et al. (2013); Durand et al. (2014), 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).

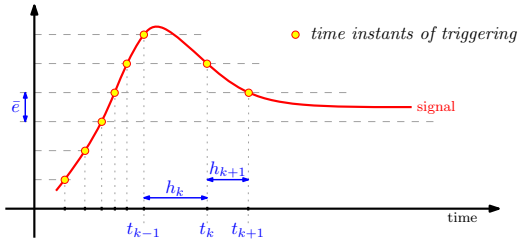
Proportional-integral-derivative (PID) controllers are the most employed controllers in industry or even in hightech devices because of their capability to provide a satisfactory performance for many processes with a relative easy design and by the availability of a large number of tuning rules. However, the growing decentralization of current processes and the number of involved computational elements demand the redesign of PID control schema. For this reason, event-triggered PID controllers were addressed by numerous researches in the last years. An original

and simple event-based PID control architecture was proposed in Årzén (1999). The suggested scheme updates the control signal only when the system output crosses a certain threshold. Whereas an event was enforced with a mix of level crossings and the use of a timer in order to bound sampling period (for stability reason) in the initial approach, this bounded period was then removed in Durand and Marchand (2009) because, in fact, the Nyquist-Shannon sampling condition is no more consistent in the asynchronous framework. Nevertheless, a safety limit is uselessly applied in reported works like in Sánchez et al. (2009b); Heemels et al. (2009); Mounier et al. (2011). Recently, different event-based PID and PI algorithms were also developed for open-loop stable first-order systems with delay, where simulations and experimental results verify the effectiveness, see Beschi et al. (2012); Ruiz et al. (2014). As evidenced by the above reviewed literature, very little attention has been dedicated to design simple and computational efficiently event-triggered PID controllers "without safety limit condition". Therefore, it is proposed here to clearly highlight the efficiency of such an approach by implementing a controller for an open-loop unstable system, that is an unmanned aerial vehicle, and testing the proposed approach in real-time.

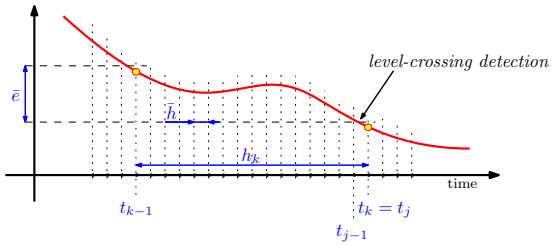
The rest of the document is organized as follows. Classical (time-triggered) PID controllers and the original event-based version from Årzén (1999) are introduced in sections 2 and 3. Event-based PID controllers without safety limit condition are then detailed in section 4. Different algorithms are presented but, whereas only the integral term was addressed in Durand and Marchand (2009) — as well as in Årzén (1999)) — both integral and derivative terms are concerned in the present paper. Furthermore, the proposed algorithms are now based on different approximation methods. A sketch of stability analysis is given in section 5. The experimental platform is depicted in section 6. The different approaches are tested for controlling the position of a real-time mini quadrotor helicopter using a motion capture system with deported controller. Experimental results highlight the capabilities of the proposed approaches in reducing the control/communication updates while maintaining similar performance. Conclusions and future works finally end the paper in section 7.

Notation: Afterwards, let $j \in \mathbb{Z}$ denote the beginning time of the current control sample in the periodic sampling scheme, that gives $t_j := j\bar{h}$, where $\bar{h}$ is the (constant) sampling period. Also, let $k \in \mathbb{Z}$ denote the beginning time of the current control sample in the non-uniform event-based sampling scheme, that is t_k, while $h_k := t_k - t_{k-1}$ denotes the (varying) sampling interval. In the present paper, an event is enforced using a level-crossing detection mechanism, that is when the signal crosses a given (relative or absolute) threshold $\bar{e}$, also called level. The different notations for the non-uniform sampling scheme are represented in Fig. 1(a).

Remark 1.1. Fig. 1(a) is intentionally not thorough since, actually, an event can only be detected periodically, at a discrete instant t_j (and not at any instant t) by construction of the event-based mechanism. Typically, if an event occurs (because a level crossing is detected) at a given time t between two sampling instants t_{j-1} and t_j, the sampling



(a) Representation of the time instants and the sampling intervals.



(b) Sequences of t_j and t_k.

Fig. 1. Event-driven control scheme.

instant for the event-based scheme really occurs at time $t_k = t_j$ (and not t), as represented in Fig. 1(b). This will be further discussed in section 3, however, next figures will not be as detailed anymore for the sake of not overload.

2. CLASSICAL (DISCRETE-TIME) PID CONTROL

Equations of the continuous-time PID controller are

$$u(t) = u_p(t) + u_i(t) + u_d(t) \tag{1}$$
$$\text{with} \quad \left| \begin{array}{l} u_p(t) = K_p e(t) \\ u_i(t) = K_i \displaystyle\int_{t_0}^{t} e(t)dt \\ u_d(t) = K_d \dfrac{de(t)}{dt} \end{array} \right.$$

where u is the control signal and e is the error between the measurement and a given reference to track, u_p, u_i and u_d are the proportional, integral and derivative parts of the PID controller, K_p, K_i and K_d are tunable parameters.

In frequency (Laplace) domain, this gives

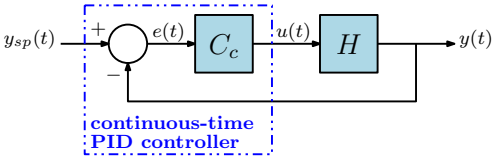
$$\left| \begin{array}{l} U_p(s) = K_p E(s) \\ U_i(s) = \dfrac{K_i}{s} E(s) \\ U_d(s) = K_d s E(s) \end{array} \right. \tag{2}$$

where s is the Laplace operator. Moreover, a low-pass filter is added in the derivative term (to avoid problems with high frequency measurement noise), which yields

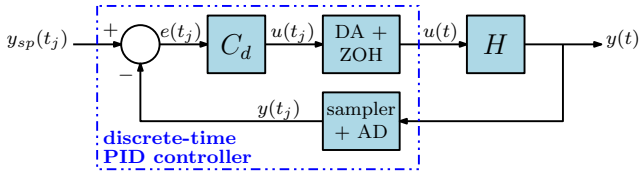
$$U_d(s) = \frac{K_d s}{1 + T_f s} E(s) \tag{3}$$

where T_f is the filter's time constant.

The control architecture is depicted in Fig. 2(a), where H is the plant to control, y its sensed output and y_{sp} a given reference to track. C_c is the continuous-time controller, derived from equations (2)-(3). It is based on the error signal $e := y_{sp} - y$ to compute the control signal u that acts on the plant input.



(a) Continuous-time PID controller.



(b) Discrete-time PID controller.

Fig. 2. Classical PID controller setups.

Fundamental differences for a discrete-time version are that i) the discrete controller operates on samples of the sensed plant output rather than on continuous signals, i.e. t_j instead of t, and ii) the dynamics are implemented by algebraic recursive equations instead of differential equations (Franklin et al. (1997)). The architecture is depicted in Fig. 2(b), where C_d is the discrete-time controller. An analog-to-digital (AD) converter changes a physical variable into a binary number (error of quantization is neglected in the present paper). The conversion occurs repetitively every constant sampling period $\bar{h}$, the sampled signal is $y(t_j)$. The reference is directly considered as a discrete signal $y_{sp}(t_j)$ here, but it could also be converted and sampled. The error is $e(t_j) := y_{sp}(t_j) - y(t_j)$. The controller then feeds the control signal $u(t_j)$, that is changed to a continuous variable $u(t)$ by a digital-to-analog (DA) converter, with a constant value throughout the sampling period $\bar{h}$ thanks to a zero-order hold (ZOH).

Algebraic recursive equations for the digital controller can be obtained making some approximations on the mapping of the s-plane in (2)-(3) to the z-plane. The resulting algorithm in discrete-time domain is

$$u(t_j) = u_p(t_j) + u_i(t_j) + u_d(t_j) \tag{4}$$

where t_j denotes the (periodic) sampling instants.

Remark 2.1. The control signal (4) is sent to the actuator of the plant as soon as it is available, assuming the time delay is minimized by making the calculations of the control law as short as possible (Åström and Murray (2008)). This explains why t_j can be in both left and right sides of formulas in the sequel.

The proportional part is easily straightforward from (2)

$$u_p(t_j) = K_p e(t_j) \tag{5}$$

However, several solutions exist for the integral and derivative parts. In this section, first-order approximation methods are applied. Three cases are treated in particular. More complex approximation methods could be applied (but this is not the aim of the present paper).

2.1. The forward difference approximation (or *explicit Euler's method*) is based on the first-order Taylor series expansion given by $s = \dfrac{1 - z^{-1}}{\bar{h} z^{-1}}$.

The forward approximation gives

$$\left| \begin{array}{l} u_i(t_j) = u_i(t_{j-1}) + K_i \bar{h} e(t_{j-1}) \\ u_d(t_j) = \dfrac{T_f - \bar{h}}{T_f} u_d(t_{j-1}) + \dfrac{K_d}{T_f} \big[e(t_j) - e(t_{j-1}) \big] \end{array} \right. \tag{6}$$

2.2. The backward difference approximation (or *implicit Euler's method*) is based on the first-order Taylor series expansion given by $s = \dfrac{1 - z^{-1}}{\bar{h}}$.

The backward approximation gives

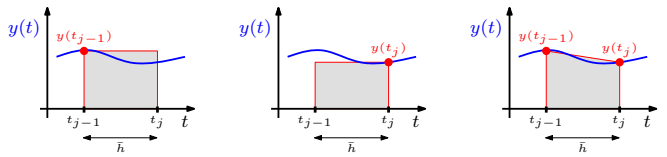
$$\left| \begin{array}{l} u_i(t_j) = u_i(t_{j-1}) + K_i \bar{h} e(t_j) \\ u_d(t_j) = \dfrac{T_f}{T_f + \bar{h}} u_d(t_{j-1}) + \dfrac{K_d}{T_f + \bar{h}} \big[e(t_j) - e(t_{j-1}) \big] \end{array} \right. \tag{7}$$

2.3. The bilinear approximation (or *Tustin's method*), based on the trapezoidal rule, is a first-order approximation of the natural logarithm function, that is an exact mapping of the z-plane to the s-plane: $s = \dfrac{2}{\bar{h}} \dfrac{1 - z^{-1}}{1 + z^{-1}}$.

The bilinear approximation gives

$$\left| \begin{array}{l} u_i(t_j) = u_i(t_{j-1}) + \dfrac{K_i \bar{h}}{2} \big(e(t_j) + e(t_{j-1}) \big) \\ u_d(t_j) = \dfrac{2T_f - \bar{h}}{2T_f + \bar{h}} u_d(t_{j-1}) + \dfrac{2K_d}{2T_f + \bar{h}} \big[e(t_j) - e(t_{j-1}) \big] \end{array} \right. \tag{8}$$

These three first-order approximation methods are illustrated in Fig. 3 for the integral part computing. Note that the different approximations work for smooth signals and are more accurate for smaller sampling period $\bar{h}$. This is why some improvements will be mandatory in a non-uniform sampling scheme where the (varying) sampling interval can increase. This will be further explained in section 3.



(a) Forward approximation. (b) Backward approximation. (c) Bilinear approximation.

Fig. 3. Comparison of three first-order numerical methods of approximation used for discretization.

Remark 2.2. Whereas the backward and bilinear methods are numerically stable, the forward one can be numerically unstable for some $\bar{h}$: the numerical solution can grow very large for equations where the exact solution does not, especially for stiff equations. Intuitively, this is because the forward approximation only considers the last value of the signal to integrate (and not the current one). For this reason, the forward method should not be a good solution when the sampling interval can become huge.

Remark 2.3. The bilinear method reduces the error of approximation but is more complex in return. This is why this method will not be detailed in the sequel. Nevertheless, an extension to the techniques proposed afterwards is trivial.

3. EVENT-BASED PID CONTROL

3.1 Årzén's event-based PID controller

An original event-based PID controller was proposed for the first time in Årzén (1999). The basic setup consists in two parts: a time-triggered event detector and an event-triggered PID controller (C_{eb}), as depicted in Fig. 4.

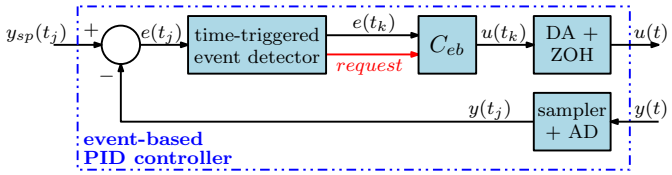


Fig. 4. Event-based PID controller setup.

Time-triggered event detector:

An event detector is used for level-crossing detection. This first part is periodically sampled with the period $\bar{h}$ (that is the same as for the corresponding conventional time-triggered PID). Typically, it runs like a zero-order holder. However, the holding instants (afterwards called events) are non uniform in time and decided with respect to the input dynamics. Therefore, the aim of the event logic is to determine the events. Then, the output holds the input signal constant between two successive events. A request is also generated at each event to trigger the second part (i.e. the event-triggered PID controller).

In the original Årzén's setup, an event occurs:

i) either when the relative measurement crosses a given level $\bar{e}$, that is when

$$\left\| e(t_j) - e(t_{k-1}) \right\| \geq \bar{e} \tag{9}$$

ii) or if the maximal sampling period is achieved, that is when

$$t_j - t_k \geq h_{max} \tag{10}$$

This second condition was added in order to ensure stability by fulfilling the Nyquist-Shannon sampling condition. Furthermore, as noticed in section 2, the different methods of approximation are more accurate for a smaller sampling period. In this sense, h_{max} allows to limit the increase of the sampling interval $h_k := t_k - t_{k-1}$.

Remark 3.1. Conditions (9) and (10) are only verified at a periodic instant t_j (since the event detector is time triggered with a constant sampling period).

Event-triggered PID controller:

The second part calculates and updates the control signal only when a request is received from the first part, that is when an event was detected by the event detector. The length of the (varying) sampling intervals h_k is defined by two successive events. The control signal is kept constant during the interval h_k.

For the event-based PID control law, the forward and backward difference approximation methods are applied for the integral and derivative parts respectively in Årzén (1999), that gives

$$u_i(t_{k+1}) = u_i(t_k) + K_i h_k e(t_k) \tag{11}$$

$$u_d(t_k) = \frac{T_f}{T_f + h_k} u_d(t_{k-1}) + \frac{K_d}{T_f + h_k} \left[e(t_k) - e(t_{k-1}) \right] \tag{12}$$

where the varying interval h_k replaces the constant period $\bar{h}$ in the conventional equations (6) and (7).

Remark 3.2. Instead of calculating the integral part at the current time $u_i(t_k)$ based on the previous value of the signal $e(t_{k-1})$ — as this was the case in (6) — in (11) it is pre-calculated at the current sampling time t_k for the next sample (in order to reduce the control delay) using the current value of the signal $e(t_k)$. Then, the next integral part $u_i(t_{k+1})$ will be used in the PID control law (4) at the next event t_{k+1}. This is possible because the change of the error remains lower than $\bar{e}$ between two successive events by construction, thanks to condition (9).

In the sequel, the initial Årzén's proposal is firstly improved correcting a forgetfulness in the discretization of the integral part (see section 3.2). Then, the *safety limit condition* (10) is removed and the event-based PID controller is modified in consequence (see section 4), changing either the event condition (that is the way to generate an event in order to calculate a new control signal) or the control equations (adapting the conventional control strategy to an event-driven framework).

3.2 Discretization improvement for the integral part

The method of discretization has to be carefully selected in a non-uniform sampling interval scheme. For instance, a forgetfulness was done in the original Årzén's approach, as noticed in Durand and Marchand (2009). Indeed, it is shown in Fig. 5(a) that the forward difference approximation needs the next sampling interval h_{k+1} to calculate the next integral part, whereas the current interval h_k is applied in the Årzén's mix up (11). Actually, the correct (non-uniform sampling) forward equation should be

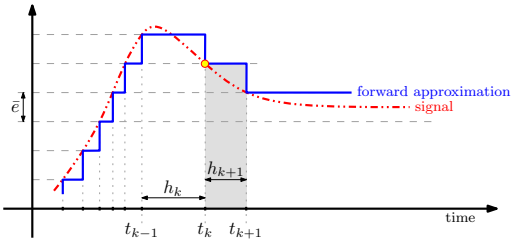
$$u_i(t_{k+1}) = u_i(t_k) + K_i h_{k+1} e(t_k) \tag{13}$$

However, the next value h_{k+1} is not *a priori* known (since it varies). If one still wants to use the forward approximation, a solution may be to post-calculate the current integral part by shifting the time instants in (13), that gives $u_i(t_k) = u_i(t_{k-1}) + K_i h_k e(t_{k-1})$. But the interest in reducing the control delay is no more reached and a old value of the error is applied. Furthermore, the forward approximation is not numerically stable (as noticed in Remark 2.2).

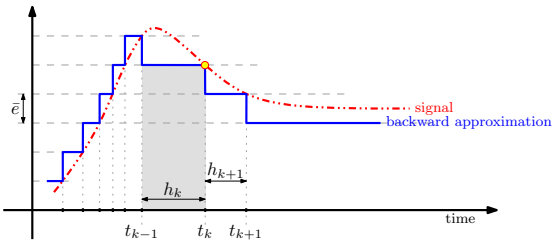
Calculating the integral part with a more recent value of the error seems a better solution. For this reason, the backward difference approximation is generally preferred. Then, the integral part is calculated with the current sampling period h_k and error $e(t_k)$, as represented in Fig. 5(b). From (7), the (non-uniform sampling) backward equation for the integral part is

$$u_i(t_k) = u_i(t_{k-1}) + K_i h_k e(t_k) \tag{14}$$

Remark 3.3. In order to simplify the understanding, Fig. 5(a) and Fig. 5(b) illustrate idealized behaviors of



(a) Forward approximation.



(b) Backward approximation.

Fig. 5. Comparison between forward and backward approximations in the non-uniform sampling scheme.

the event-based scheme since they do not consider the synchronization of the events with the event detector sampling instants t_j. Indeed, the time of level crossing could be between two successive time-triggered sampling instants, that are $t_j - \bar{h}$ and t_j, because the event detector is time driven with the constant sampling period $\bar{h}$ by construction (see section 3.1). Nevertheless, the right behavior will be discussed and taken into account in the sequel.

4. ANALYSIS AND MODIFICATION OF THE EVENT-BASED SCHEME

In order to simplify the original Årzén's event-based setup, the safety limit condition (10) — initially introduced for stability reason — is removed here. However, several modifications are mandatory with such a simplification.

4.1 Boundary of the integration increment

Fig. 6 depicts an example of the error e between the measured signal y (i.e. output of the controlled system) and a given reference to track y_{sp}, in order to visualize how it behaves during transient and steady-state intervals (note that the error signal is zoomed but proportions are respected). Typically, many events should occur when the measured signal is converging to the reference (i.e. during transients) since the error is varying. Conversely, no event should be enforced once the measured signal vanishes close to the reference (i.e. during steaty-state intervals). By removing the safety condition (10), the duration of a steady-state interval can increase as far as the error does not cross the threshold level $\bar{e}$ — because of (9) — whereas it was limited to h_{max} before. This allows to reduce even more the control updates but new problems have to be considered, in particular when the reference drastically changes after a large steady-state interval. This can lead to overshoots whereas the system output is over-corrected in case of huge sampling interval h_k and/or huge error $e(t_k)$. Such a case is deeply analyzed here.

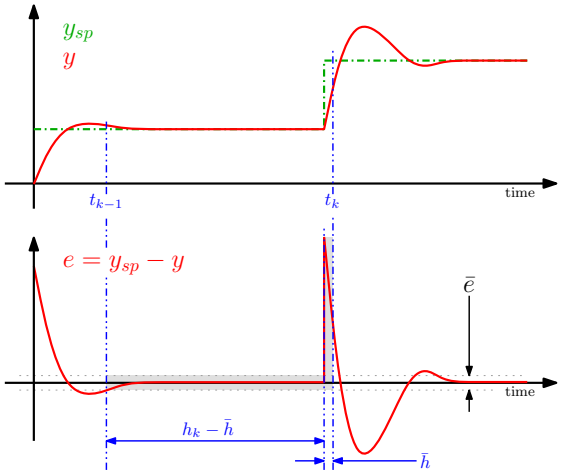


Fig. 6. Boundary of the integration increment between two successive events after a large steady-state interval in the non-uniform sampling scheme.

Actually, issues come from the calculation of the integral part (14) that involves the product $h_k e(t_k)$ — afterwards called the *integration increment* Γ_k. Indeed, the integration increment is over-estimated in the current version of the control algorithm. However, in fact the time interval between two successive events after a large steady state can be divided into two parts, as drawn in Fig. 6, that are i) the time interval where the signal is really in a steady state and ii) the time interval required to detect a new level crossing.

i) The first part starts the last time a control signal was computed, that is at the sampling instant t_{k-1}, and finishes just before the reference changes. Because of the time-triggered event detector, this time is $t_k - \bar{h}$. During this interval, i.e. $h_k - \bar{h}$, the error remains very small, it is lower than the detection level $\bar{e}$ else this is not the steady state.

ii) Then, because the error becomes higher than the detection level $\bar{e}$, the second part starts. A request is sent and a new control signal is finally calculated at time t_k. This instant does not necessarily occurs exactly when the reference changes because of the periodic behavior of the time-triggered event detector (see Fig. 1(b) and Remark 3.1), nonetheless the interval to detect the level crossing is bounded by the sampling period $\bar{h}$.

To summarize, the time interval between two successive events can be divided into i) a first part where *the sampling interval increases but the error remains small* and ii) a second part where *the error could become large but only during a short instant*. Taking into account such considerations makes the integration increment $\Gamma_k := h_k e(t_k)$ will not explode anymore, since both $h(\cdot)$ and $e(\cdot)$ compensate themselves each other. Finally, the integral part (14) is hence bounded in practice

$$u_i(t_k) = u_i(t_{k-1}) + K_i \Gamma_k \qquad (15)$$
$$\text{with} \quad \Gamma_k \leq (h_k - \bar{h})\bar{e} + \bar{h} e(t_k)$$

and this boundary has to be integrated in the event-based control algorithms without safety limit condition that are then developed in section 4.5 based on that result.

Furthermore, note that the inequality (15), which was initially built for the steady-state intervals, is finally correct for the whole running. Indeed, (15) becomes (14) during a transient phase (where an event is enforced at each time-triggered sampling period) since $h_k = \bar{h}$ in this case.

4.2 Sampling interval in the derivative term

The derivative part of the event-based PID controller can be canceled when removing the safety limit condition (10), because a huge sampling interval h_k makes vanishing (12).

In practice, the derivative term calculates the slope of the error over time in order to predict the system behavior. As a consequence, if this rate of change vanishes, the derivative action will not be applied anymore. For this reason, a modification is also required. The two parts of the time interval between two successive events after a large steady state (see Fig. 6) are also analyzed here:

i) During the first part, the time derivative of the error remains small because the error is lower than $\bar{e}$, the reference does not change and the control is kept constant. The only reason to have an important derivative is in case of perturbation, but the perturbation is assumed to be small enough (else it will enforce a new event). For these reasons, the derivative term can be neglected in the first part of the steady state.
ii) During the second part, the time derivative of the error becomes really important because of the reference change. Nevertheless, whereas h_k is considered in the existing approach (12), only the interval $\bar{h}$ can be considered because, in practice, the rate of change can only occur during the level detection interval.

To summarize, only the last sampling period $\bar{h}$ is taken into account in the derivative part, which hence remains as defined in (7) for the backward case.

4.3 Forgetting factor of the sampling interval

Another solution when removing the safety limit condition (10) consists in adding a forgetting factor of the sampling interval so that, after a long steady-state interval, the value of h_k is reduced enough to not impact the control signal too much. This is true for both the integral as well as the derivative parts. The idea is to replace the linearly varying sampling interval h_k in (14) and (12) by an exponentially decreasing one. The approach proposed here is somehow similar to the anti-windup mechanism used in control theory, where the error induced by the saturation has to be compensated.

For the derivative term, an exponential function is chosen such that the impact of the sampling interval decreases as the elapsed steady-state time increases, that is

$$h^d_{exp}(h_k) = \bar{h} + (h_k - \bar{h})e^{-\alpha_d(h_k-\bar{h})} \tag{16}$$

where $\alpha_d > 0$ is a degree of freedom to increase/decrease the exponential sampling interval (16). Such a forgetting factor allows to have $h^d_{exp}(h_k)$ close to the sampling interval value h_k when it is small, whereas only the last sampling period $\bar{h}$ is considered when it is large (as suggested in section 4.2).

For the integral term, an exponential function is chosen such that

$$h^i_{exp}(h_k) = h_k e^{-\alpha_i(h_k-\bar{h})} \tag{17}$$

where $\alpha_i > 0$ is a degree of freedom to increase/decrease the exponential sampling interval (17). This forgetting factor allows to have $h^i_{exp}(h_k)$ close to the sampling interval value h_k when it is small, whereas it exponentially vanishes when h_k increases.

The dynamics of (16) and (17) with respect to h_k is depicted in Fig. 7.

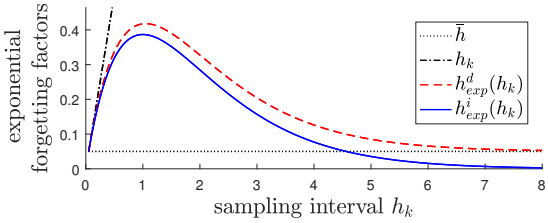


Fig. 7. Evolution of the exponential forgetting factor with respect to the sampling interval, for $\bar{h} = 0.05\,s$ and $\alpha_i = \alpha_d = 1\,s^{-1}$.

4.4 Level-crossing mechanism

Årzén suggested to calculate and update the control signal using a *relative* measurement (9), that is when the error changes enough from its last value, i.e. $\left\|e(t_k) - e(t_{k-1})\right\| \geq \bar{e}$ (see section 3.1). However, such a setup creates troubles because the system can reach a final (undesired) value which is different of the reference. In this case, no event occurs anymore whereas the system is not stabilized. For this reason, an *absolute* measurement is preferred here as in Durand and Marchand (2009). A new control signal is thus calculated as soon as the current error crosses the detection level, that is when

$$\left\|e(t_j)\right\| \geq \bar{e} \tag{18}$$

Remark 4.1. With the proposed method, the number of samples should inevitably increase during the transients but, at least, the error between the system output and the reference will be lower than $\bar{e}$ during the steady-state intervals. This was not the case before. In fact, this modification was not required in the original Årzén's setup because the system always reached the reference thanks to the safety limit condition (10).

Remark 4.2. Whereas the level-crossing detection mechanisms in (9) and (18) are different (relative vs. absolute), the meaning of the constant $\bar{e}$ is the same: it is a threshold used to enforce the events. Therefore, the setup detailed before, in particular in Fig. 6, is still true.

4.5 Event-based PID control without safety limit condition

Based on the suggested modifications above, different strategies are proposed: the first one where nothing else is done than removing the safety limit condition (10), and the other ones where the integral and/or derivative parts of the control law are modified in order to reduce their impact after a large steady-state interval.

Algorithm 1 – Only without safety limit condition:

This case corresponds to the Årzén's proposal where the safety limit condition (10) is removed without doing anything else. Important overshoots are expected because of the principle described in section 4.1.

Remark 4.3. This first algorithm is presented in order to show that the modifications which follow are required.

Algorithm 2 – Saturation of the integration increment:

This strategy consists in bounding the integration increment Γ_k after a large steady-state interval in order to reduce the overshoots (as discussed in section 4.1). The product between $h(\cdot)$ and $e(\cdot)$ is thus saturated according to (15), which yields

$$u_i(t_k) = u_i(t_{k-1}) + K_i\Gamma_{sat}(t_k) \tag{19}$$
$$\text{with} \quad \Gamma_{sat}(t_k) = (h_k - \bar{h})\bar{e} + \bar{h}e(t_k)$$

Only the last sample is taken into account for the derivative term, as defined for the classical approach in (7) (see section 4.2 for further details).

Algorithm 3 – Exponential forgetting factor of the sampling interval:

A forgetting factor of the sampling interval is added in such a way it does not impact the control signal too much after a long steady state (as discussed in section 4.3). The exponentially decreasing sampling intervals $h^d_{exp}(h_k)$ in (16) and $h^i_{exp}(h_k)$ in (17) are applied in the derivative and integral parts respectively. This gives

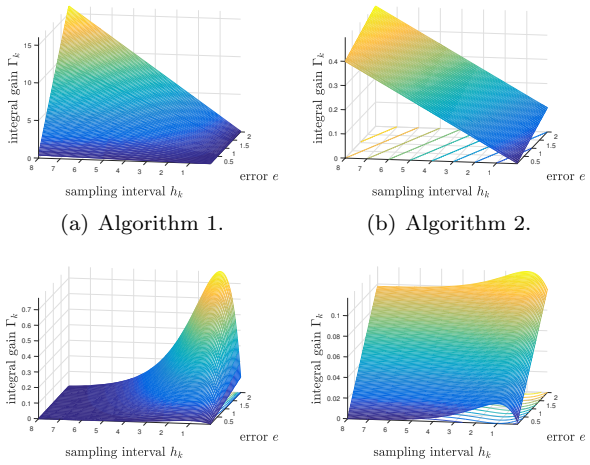
$$u_i(t_k) = u_i(t_{k-1}) + K_i\Gamma_{exp}(t_k) \tag{20}$$
$$\text{with} \quad \Gamma_{exp}(t_k) = h^i_{exp}(h_k)e(t_k)$$
$$u_d(t_k) = \frac{T_f}{T_f + h^d_{exp}(h_k)}u_d(t_{k-1}) + \frac{K_d}{T_f + h^d_{exp}(h_k)}\big[e(t_k) - e(t_{k-1})\big] \tag{21}$$

Algorithm 4 – Hybrid strategy:

This is a mix of the previous algorithms. In fact, in the first case the integration increment increases with respect to $h(\cdot)$ and $e(\cdot)$, as drawn in Fig. 8(a). The second strategy minimizes the impact of the integration increment but it still increases with respect to both $h(\cdot)$ and $e(\cdot)$, as shown in Fig. 8(b). Finally, the third algorithm adds an exponential forgetting factor of the sampling interval in such a way that the integration increment decreases but, as a result, the product between $h(\cdot)$ and $e(\cdot)$ is higher when the sampling interval is small, as represented in Fig. 8(c). The idea here is to have a small impact of the sampling interval all the time. For this reason, it is proposed to apply the exponential forgetting factor into the algorithm with saturation, which leads

$$u_i(t_k) = u_i(t_{k-1}) + K_i\Gamma_{hybrid}(t_k) \tag{22}$$
$$\text{with} \quad \Gamma_{hybrid}(t_k) = \big(h^i_{exp}(h_k) - \bar{h}\big)\bar{e} + \bar{h}e(t_k)$$

The evolution of the integration increment of this "hybrid" algorithm is represented in Fig. 8(d).



(a) Algorithm 1. (b) Algorithm 2.

(c) Algorithm 3. (d) Algorithm 4.

Fig. 8. Evolution of the integration increment for the different algorithms, with $\bar{h} = 0.05\,s$, $\bar{e} = 0.01$ and $\alpha_i = 1\,s^{-1}$.

On the other hand, the exponential forgetting factor of the sampling interval is applied in the derivative term, as previously defined in (21).

4.6 Simulation results

Consider a continuous-time second-order system in its general form

$$H(s) = \frac{\kappa}{1 + \dfrac{2\zeta}{\omega_n}s + \dfrac{1}{\omega_n^2}s^2} \tag{23}$$

where κ is the static gain, ζ is the damping ratio and ω_n is the natural frequency. Consider also the PID controller in (2) and the desired closed-loop system as a first-order system of the form

$$H_{cl}(s) = \frac{1}{1 + \tau s} \tag{24}$$

for a given time constant τ. Then, a naive solution (using pole compensation) for the control parameters is

$$K_p = \frac{2\zeta}{\kappa\tau\omega_n}, \quad K_i = \frac{1}{\kappa\tau}, \quad K_d = \frac{1}{\kappa\tau\omega_n^2} \tag{25}$$

whereas the filter's time constant T_f in (3) is chosen with respect to the value of the sampling period $\bar{h}$.

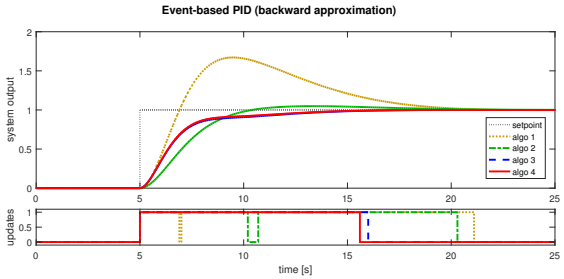
Remark 4.4. The values of the control parameters (25) are obtained by pole placement of the closed-loop system considering the time-triggered PID controller. Then, the event-based PID controllers are designed with these same values. Note that the aim here is to find the best control synthesis (otherwise pole compensation would clearly not be chosen), but to compare the proposed event-based approaches with the classical time-triggered one. In other words, the event-based scheme tries to be as close as possible of the time-triggered closed-loop shaping.

Remark 4.5. The control parameters are computed in the continuous-time domain in (25). Another solution could be to directly calculate them in discrete-time domain, this is detailed in Durand and Guerrero-Castellanos (2015).

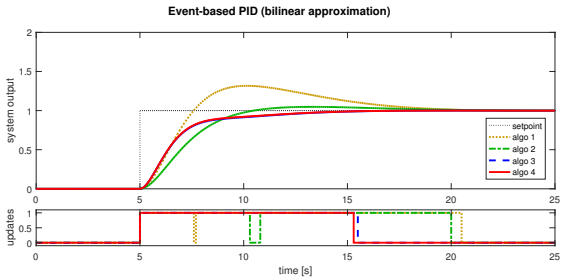
Consider the application case above, with:

- The continuous-time second-order system (23), where $\kappa = 0.2$, $\zeta = 0.8$ and $\omega_n = 0.5\,s^{-1}$;
- The desired closed-loop system as a first-order system of the form (24), where $\tau = 1.98\,s$;
- The sampling period $\bar{h} = 0.1\,s$;
- The PID control parameters are as defined in (25) (using pole compensation), that gives $K_p = 8.08$, $K_i = 2.53\,s^{-1}$, $K_d = 10.11\,s$ and $T_f \approx 5\bar{h}$;
- The exponential decay $\alpha_i = \alpha_d = 10\,s^{-1}$ and detection limit $\bar{e} = 0.01$ in the event-based controllers.

The different event-based PID algorithms without safety limit condition are compared in Fig. 9 for both backward and bilinear approximation methods. Note the reference changes (from 0 to 1) at time $t = 5\,s$. On the one hand, the frequency of updates is reduced in all event-based cases compared to the classical time-triggered strategy. This is depicted in the (extra) bottom plot, where '1' means the control law is calculated and updated during the sampling period $\bar{h}$, '0' means the control is kept constant. Furthermore, when the safety limit condition (10) is removed (algorithm 1), an important overshoot occurs after a large steady-state interval as expected. Such an overshoot is reduced when saturating the integration increment (algorithm 2), applying an exponential forgetting factor of the sampling interval (algorithm 3) or mixing both latter strategies (algorithm 4). On the other hand, the bilinear approximation method in Fig. 9(b) gives better results than the backward one in Fig. 9(a) in terms of frequency updates and tracking performance (but with a little bit more complex control algorithm in return).



(a) Backward approximation.



(b) Bilinear approximation.

Fig. 9. Simulation results: closed-loop system responses with event-based PID controllers using the different algorithms without safety limit condition.

Finally, the hybrid algorithm 4 with bilinear approximation is compared with the Årzén's algorithm in Fig. 10 (the original Årzén's algorithm was modified in order to also implement the bilinear approximation). However, whereas only absolute measurement (18) enforces events in the proposal case, events are enforced either with the relative measurement (9) or when the safety limit condition (10) is satisfied with Årzén. On the one hand, it can be noticed periodic events because of such a safety limit condition. On the other hand, the proposal only generates events during transients and then no event during steady states.

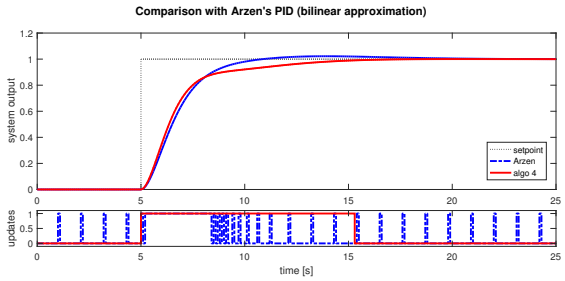


Fig. 10. Simulation results: comparison with Årzén's event-based PID controller (bilinear approximation).

5. STABILITY ANALYSIS

Only a sketch of analysis is given here, because the principle is trivial: if the system becomes unstable due to the fact that between two successive events it runs in open loop, then the dynamic of the error activates the event condition and, hence, the system is driven in a stable regime. In other words, the event-based scheme consists in generating events when the system becomes unstable (that is when the error grows), while keeping the control signal constant when the system is stable (when the measure is closed to the reference value). The closed-loop stability is based on the stability property of the PID controller (assuming the control parameters lead to a stable closed-loop system). Note that the controller runs in open loop between two successive events (e.g. t_{k-1} and t_k), but in fact it is a closed-loop system thanks to the event detector which periodically monitors the system output (at each periodic sampling time t_j) and decides when to update the control signal. In the worst case (in term of frequency of events), the event-based feedback works as when applying the classical discrete-time controller: the events occur periodically, with $\bar{h}$ as the sampling period (the minimum sampling time by construction). Conversely, when the interval between two successive events increases, stability is still ensured because the error remains lower than $\bar{e}$ by construction, thanks to the triggering condition (18), then allowing a practical stability.

6. EXPERIMENTAL RESULTS: APPLICATION TO A MINI QUADROTOR HELICOPTER

Unmanned aerial vehicles (UAVs), and particularly the *mini quadrotor helicopters*, give rise to great enthusiasm in research because of its high manoeuvrability, its payload capacity and its ability to hover, see Castillo et al. (2004). However, the quadricopter is nonlinear, unstable and under-actuated with only four input forces (voltage of each rotor) for six output coordinates to control (roll ϕ, pitch θ and yaw ψ for the attitude; x, y and z for the position). Fortunately, its model can be broken down into

two subsystems: one defining the translation movement (position) and the other one the rotation movement (attitude). Both are coupled in cascade since the translational subsystem depends on the rotational one, but the rotational subsystem is independent of the translational one.

The algorithms proposed in the present document will be tested in practice for such a control architecture. Event-based control is quite new for UAVs systems, since the system has to be actively actuated to remain stable. Nevertheless, attitude control was addressed in Téllez-Guzmán et al. (2012); Guerrero-Castellanos et al. (2013, 2014) and position control is now addressed here.

6.1 Experimental platform

The system is a 18 grams Blade *Nano QX* quadricopter (see Fig. 11). In order to release the platform from decision making related to guidance and navigation, position and orientation are calculated by a *Vicon* motion capture system, where the movement of the vehicle is processed through *T40s* high-resolution cameras (based on the principle of inverse projection and triangulation). Measurements are sent to the control unit through a UDP frame every 2ms. Control algorithms are programmed in *Matlab/Simulink* and implemented in the control unit side in real time at 200Hz to a target computer using *xPC target* toolbox. Finally, control variables are sent to the quadricopter through a built-in bridge that converts UDP frames to DSMX 2.4Ghz protocol. An overview of this architecture is presented in Fig. 12.

Fig. 11. Representation of the different control variables in the mini quadrotor helicopter.

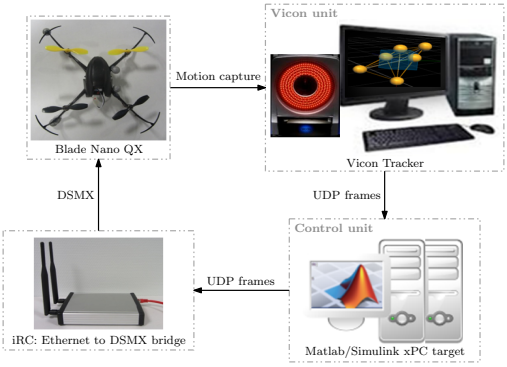


Fig. 12. Experiment architecture.

The leveling control of the mini quadricopter (the control of pitch θ and roll ϕ) is done in a (non accessible) internal loop. Only the control of x, y, z positions and yaw ψ angle is possible. It is assumed that all the control variables can be independently controlled. It is also assumed that both pitch and roll similarly behave (thanks to the symmetry of the quadcopter) and, consequently, x and y positions too. The aim of the control strategy is to stabilize the x and y positions of the quadrotor helicopter to a given reference. The yaw angle ψ and altitude z are not addressed here and will be constant. In other words, the control objective consists in calculating the angles ϕ and θ as well as the thrust T (all needed by the internal loop) that will drive the quadcopter to a given position x and y.

6.2 System model for position control

Let consider the dynamical system model

$$m\ddot{\xi} = m\boldsymbol{g} + RT\boldsymbol{b}_3 \tag{26}$$

where $\xi = (x\ y\ z)^T$, m is the mass, $\boldsymbol{g} = [0\ 0\ g]^T$ with g the gravitational constant, T is the thrust of the quadricopter. $\boldsymbol{b}_3$ is a unit vector of the body fixed frame and R is the rotation matrix to express the orientation of the body fixed frame in the inertial frame

$$R = \underbrace{\begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_z} \underbrace{\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}}_{R_y} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}}_{R_x}$$

Developing these equations gives the model

$$m\ddot{x} = T(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi) \tag{27}$$

$$m\ddot{y} = T(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi) \tag{28}$$

$$m\ddot{z} = T\cos\theta\cos\phi - mg \tag{29}$$

6.3 From nonlinear to linear control

The tricky idea consists in dividing the controller into two cascaded parts, as suggested in Hably et al. (2006): a first one that contains a linear control law (i.e. a PID controller here) and a second part used to compensate for the nonlinearities of the system. The second part is calculated in such a way that the model obtained from its input to the output of the system behaves like a simple and linear system (i.e. a double integrator here, as detailed below). Therefore, the PID controller acts on the (virtual) double integrator system, while the whole controller (i.e. both PID and nonlinear parts) drives the real system.

In practice, reformulating (29) gives

$$T\cos\theta\cos\phi = m(\ddot{z} + g) \tag{30}$$

It is assumed that the quadrotor helicopter will not flip in order to avoid a situation where $\phi = \pm\pi/2$ or $\theta = \pm\pi/2$. Multiplying (27) and (28) by $\sin\psi$ and $\cos\psi$ respectively, and dividing the difference by (30) gives the relation for ϕ. Similarly, multiplying (27) and (28) by $\cos\psi$ and $\sin\psi$ respectively, and dividing the sum by (30) gives the relation for θ. The relation for T is easily obtained from (30). This yields

$$\phi = \tan^{-1}\left(\cos\theta\frac{\ddot{x}\sin\psi - \ddot{y}\cos\psi}{\ddot{z} + g}\right)$$

$$\theta = \tan^{-1}\left(\frac{\ddot{x}\cos\psi + \ddot{y}\sin\psi}{\ddot{z} + g}\right)$$

$$T = m\frac{\ddot{z} + g}{\cos\theta\cos\phi}$$

Then choosing

$$\phi = \tan^{-1}\left(\cos\theta \frac{r_x \sin\psi - r_y \cos\psi}{r_z + g}\right) \quad (31)$$

$$\theta = \tan^{-1}\left(\frac{r_x \cos\psi + r_y \sin\psi}{r_z + g}\right) \quad (32)$$

$$T = m\frac{r_z + g}{\cos\theta\cos\phi} \quad (33)$$

with r_x, r_y and r_z as virtual control inputs from the position errors on x, y and z axes respectively. Therefore, the (virtual) system to control can be written as three independent double integrators

$$\ddot{\xi} = r \quad (34)$$

where $r = (r_x \ r_y \ r_z)^T$. Eventually, considering the attitude control dynamics on the x and y accelerations as a first-order system leads to the transfer functions

$$x = \frac{\omega_x}{s^2(s+\omega_x)} r_x \quad (35)$$

$$y = \frac{\omega_y}{s^2(s+\omega_y)} r_y \quad (36)$$

$$z = \frac{1}{s^2} r_z \quad (37)$$

To summarize, the position control of the quadcopter consists in controlling three independent (filtered) double integrators (35)-(37). This will be fulfilled by a PID controller. Then, nonlinear equations (31)-(33) — i.e. the second part of the controller — allow to calculate the variables needed for the orientation control.

Remark 6.1. The altitude z is actually a double integrator system which gain decreases with respect to the battery load. This is no more detailed in the sequel since only the x and y positions are addressed, nevertheless the battery will be charged before each experiment to reduce its impact.

Remark 6.2. Since it is assumed that the drone similarly behaves in x and y axes, then the control parameters will be tuned only once and applied for both axes in the sequel.

6.4 Performance indexes

Performance indexes, introduced in Sánchez et al. (2009b), 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 (normalized and expressed in percentage);
- The IAE index, which gives information on the reference tracking (and so on the performance):

$$IAE = \sum_{j=0}^{N} \big|e(t_j)\big| dt$$

- By analogy, the IAU index gives information on the control effort:

$$IAU = \sum_{j=0}^{N} \big|u(t_j)\big| dt$$

where N is the final simulation time.

The performance indexes obtained for the different experiments (detailed below) are summarized in Table 1 for x and y positions. Results are discussed in section 6.5.

Table 1. Performance indexes for different experiments with several PID control strategies.

(a) Control of the position along x.

			Nb (%)	IAE	IAU
Classical PID	backward method		100	1.78	12.42
	bilinear method		100	1.59	12.59
Event-based PID ($\bar{e} = 10\,mm$)	backward	algo 1	94.92	1.88	12.74
		algo 2	88.46	1.86	11.94
		algo 3	92.50	1.71	11.29
		algo 4	88.61	1.79	12.22
	bilinear	algo 1	82.27	1.71	10.90
		algo 2	97.88	2.75	17.84
		algo 3	90.61	1.64	10.81
		algo 4	91.19	1.69	10.81
Event-based PID ($\bar{e} = 50\,mm$)	backward	algo 1	63.70	2.37	15.31
		algo 2	67.32	2.81	17.40
		algo 3	66.20	2.59	17.54
		algo 4	67.97	2.74	17.90
	bilinear	algo 1	63.70	2.26	13.99
		algo 2	63.89	2.79	18.01
		algo 3	59.09	2.36	15.07
		algo 4	60.01	2.44	14.61

(b) Control of the position along y.

			Nb (%)	IAE	IAU
Classical PID	backward method		100	1.55	10.29
	bilinear method		100	1.43	8.79
Event-based PID ($\bar{e} = 10\,mm$)	backward	algo 1	99.26	2.25	15.09
		algo 2	89.81	1.74	10.90
		algo 3	69.31	1.39	8.91
		algo 4	77.50	1.61	11.00
	bilinear	algo 1	78.73	1.69	10.62
		algo 2	99.26	2.73	18.67
		algo 3	78.58	1.54	9.68
		algo 4	81.43	1.50	8.85
Event-based PID ($\bar{e} = 50\,mm$)	backward	algo 1	61.13	2.23	13.53
		algo 2	64.66	2.73	16.84
		algo 3	63.70	2.72	17.85
		algo 4	65.39	2.66	17.52
	bilinear	algo 1	58.43	2.22	13.02
		algo 2	64.01	2.68	16.81
		algo 3	55.90	2.32	14.17
		algo 4	55.01	2.04	12.36

6.5 Experimental results

The mini helicopter has to track three points in the space, which coordinates are $(x, y, z) = (0, 0, 1)$, $(0, 1, 1)$, and $(1, 1, 1)$ respectively. The altitude is kept the same (i.e. $1\,m$ high) for the three points because the study only focuses on x and y position control (z is not depicted in the plots). Typically, the quadricopter has to draw a triangle in the xy-plane. The system goes to one point to the following point after a waiting time of $10\,s$ in order to analyze both the reference tracking and the stabilization. This gives different steps at different instants, which are filtered in order to avoid abrupt changes in the position references.

The system trajectory in the xy-plane are compared for several strategies, for both backward and bilinear approximation methods:

- Classical (time-triggered) PID controllers;
- Event-based PID controllers without safety limit condition, where the control signal is calculated and updated only when the system output crosses a given level $\bar{e}$:

- With saturation of the integration increment (algorithm 2);
- With exponential forgetting factor of the sampling interval (algorithm 3);
- With hybrid strategy (algorithm 4).

The control parameters (obtained by trial and error) are $K_p = 3$, $K_i = 0.1\,s^{-1}$ and $K_d = 2\,s$. The sampling period is $\bar{h} = 0.01\,s$, the derivative filter's time constant is $T_f \approx 2\bar{h}$. The event-based control parameters are $\alpha_i = \alpha_d = 1\,s^{-1}$ and $\bar{e} = 10$ or $50\,mm$. Note that the battery quadrotor is charged before each new experiment.

Experimental results are represented in Fig. 13 for the time-triggered PID control when using either the backward or the bilinear approximation methods, for a $26\,s$ interval time interval. The top plot shows the reference and the measured positions. The bottom plots show the angles (roll ϕ and pitch θ) and control signals. One can see in Fig. 13(b) that the bilinear method gives better performance than the backward one in Fig. 13(a), this is also verified when comparing the IAE and IAU indexes for both methods in Table 1.

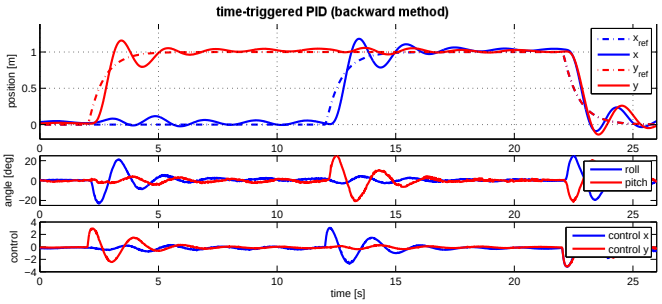
Experimental results for the event-based control are represented in Fig. 14 (only algorithm 4 with bilinear method is depicted but all of them give more or less similar results). Extra bottom plots show the sampling instants in these event-based control schemes ('1' means the control law is calculated and updated during the sampling period $\bar{h}$, '0' means the control is kept constant). In the different cases (even those not represented), the measured positions in Fig. 14(a) and performance indexes in Table 1 are close to the desired ones when the detection level is small, i.e. $\bar{e} = 10\,mm$, with a reduced frequency of updates (30% less of updates in the better case). When the detection level increases, i.e. $\bar{e} = 50\,mm$ in Fig. 14(b), the frequency of updates decreases even more (until 45% less of updates) but with weaker performance in return (because of the unstable behavior in open loop that occurs between two successive events, see section 5), so the IAE and IAU indexes increase.

To summarize, the tradeoff between performance and frequency of control updates is clearly highlighted. One can hence imagine how the computing resources utilization can be reduced in embedded systems (as well as communications in the case of a deported controller as in the present application).

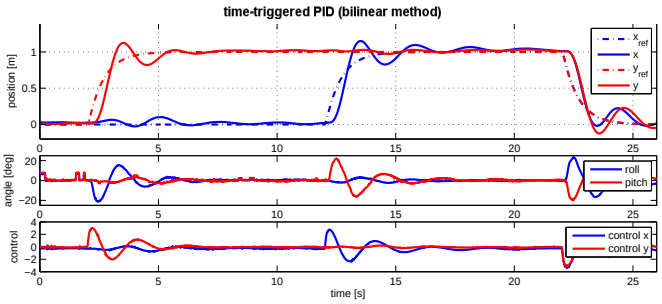
Remark 6.3. Performance can be increased with other values of the control parameters, α_i and α_d for instance, but this is not the aim of the present work.

7. CONCLUSIONS AND FUTURE WORKS

This paper recalled the classical time-triggered PID control scheme using different methods of approximation. The event-based paradigm was then introduced and event-based PID control algorithms were developed. The proposed approaches are built without safety limit condition, contrary to the original setup in Årzén (1999), and both integral and derivative terms were considered. The different approaches were tested in simulation and then on a real-time system: a mini quadrotor helicopter using a motion capture system to provide its position, where the

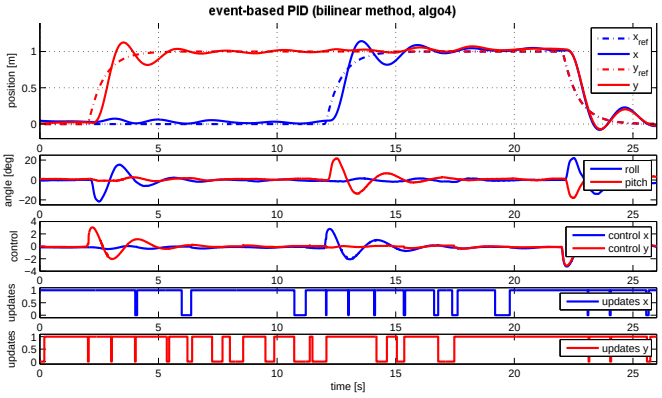


(a) Backward algorithm.

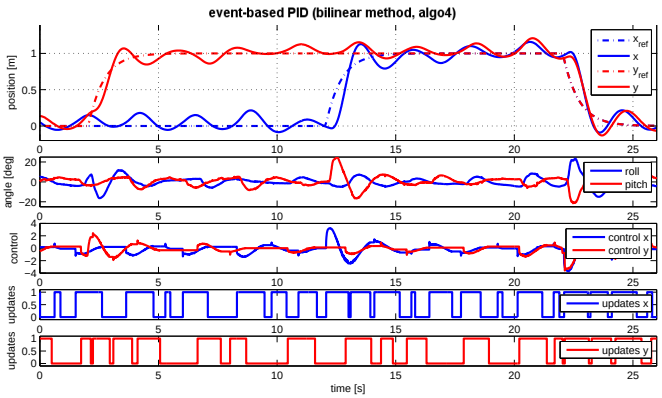


(b) Bilinear algorithm.

Fig. 13. Experimental results of the time-triggered PID controller (backward and bilinear approximation methods).



(a) $\bar{e} = 10\,mm$.



(b) $\bar{e} = 50\,mm$.

Fig. 14. Experimental results of the event-based PID controller (bilinear method, algorithm 4 for different detection levels $\bar{e}$).

controller is deported and communications are hence of high importance. Experimental results showed the effectiveness of the proposals with a reduction of the computing and 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 consider delays, as in Durand (2013). Nonlinear strategies will also be a trajectory for future works, with event-based control laws in the spirit of Marchand et al. (2013); Durand et al. (2014). 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 in reducing computation/communication resources.

ACKNOWLEDGMENT

The authors would like to thank J. Dumon and J. Minet for their technical support as well as T. Da Silva, M. Fayet, D. Manica, P. Perraud for their contribution in the context of student projects (Ense3, Grenoble university). They also would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR–11-LABX-0025). The experimental platform was designed by the GISPA-lab technical staff and partially funded by EquipEx Robotex (ANR–10-EQPX-4401).

REFERENCES

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

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

Åström, K.J. and Murray, R.M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

Beschi, M., Dormido, S., Sanchez, J., and Visioli, A. (2012). Characterization of symmetric send-on-delta PI controllers. *Journal of Process Control*, 22, 1930–1945.

Castillo, P., Dzul, A., and Lozano, R. (2004). Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Transactions on Control Systems Technology*, 12, p. 510–516.

Durand, S. (2013). Event-based stabilization of linear system with communication delays in the measurements. In *Proceedings of the American Control Conference*.

Durand, S. and Guerrero-Castellanos, J.F. (2015). Event-based digital PID control. In *Proceedings of the 1st IEEE International Conference on Event-Based Control, Communication, and Signal Processing*.

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

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

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

Franklin, G., Powell, J., and Workman, M. (1997). *Digital Control of Dynamic Systems (3rd Edition)*. Addison-Wesley.

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

Guerrero-Castellanos, J.F., Téllez-Guzmán, J.J., Durand, S., Marchand, N., Álvarez Muñoz, J., and González-Díaz, V. (2014). Attitude stabilization of a quadrotor by means of event-triggered nonlinear control. *Journal of Intelligent and Robotic Systems*, 73, 123–135.

Hably, A., Kendoul, F., Marchand, N., and Castillo, P. (2006). Further results on global stabilization of the PVTOL aircraft. In *Proceedings of the Second Multidisciplinary International Symposium on Positive Systems : Theory and Applications*, Lecture Notes in Control and Information Sciences LNCIS 341, p. 303–310. Springer.

Heemels, W., Sandee, J., and van den Bosch, P. (2009). Analysis of event-driven controllers for linear systems. *International journal of control*, 81, 571–590.

Lunze, J. and Lehmann, D. (2010). A state-feedback approach to event-based control. *Automatica*, 46, 211–215.

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

Mounier, H., Cela, A., Niculescu, S.I., and Wang, J. (2011). Event driven intelligent PID controllers with applications to motion control. In *Proceedings of the 18th world congress of IFAC*.

Ruiz, A., Jimnez, J., Sanchez, J., and S., D. (2014). A practical tuning methodology for event-based PI control. *Journal of Process Control*, 24, 278–295.

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

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

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

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

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

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