Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

HYBRID ADAPTIVE CONTROL FOR UAV DATA COLLECTION: A SIMULATION-BASED DESIGN TO TRADE-OFF RESOURCES BETWEEN STABILITY AND COMMUNICATION

Ezequiel Pecker Marcosig
Juan I. Giribet

Departamento de Ingeniería Electrónica
FI, UBA and CONICET
Av. Paseo Colón 850
C1063ACV, Buenos Aires, ARGENTINA

Rodrigo Castro

Departamento de Computación
FCEyN, UBA and ICC, CONICET
Ciudad Universitaria, Pabellón 1
C1428EGA, Buenos Aires, ARGENTINA

ABSTRACT

We present the design of a hybrid control system for an Unmanned Aerial Vehicle (UAV) used for data collection from randomly scattered wireless sensors. We postulate a restrictive scenario where a low-cost processor is in charge of both flying the UAV and resolving data communication. This raises the need for safe trade-off of computing resources between stability and throughput, adapting to an unpredictable environment (wind and sensors). We designed a strategy where a supervisory controller implements an adaptive relaxation of the sampling-period of the UAV regulation controller to favor communication tasks. To guarantee stability under period switching we update the discrete-time control law with suitable gains. The resulting system comprises continuous, discrete-time and discrete-event dynamics, including event-based adaptation of the discrete-time controller. We relied on the DEVS modeling and simulation framework to support a full simulation-based design, verification and validation process, featuring a seamless composition of the underlying hybrid domains.

1 INTRODUCTION

Hybrid feedback loops are pervasive in Cyber-Physical Systems (CPS) that integrate algorithmic and physical domains, as computational and physical processes influence each other. The uninterrupted operation of CPS under quality constraints is a key requirement for the discipline of hybrid control. The overall quality of operation for a hybrid CPS must be guaranteed in the context of unpredictable dynamic environments including resource constraints. This can be achieved by orchestrating several self-adaptive capabilities of the CPS, under the responsibility of modularized cooperative controllers. Yet, the very notions of quality can differ drastically across domains. Core algorithmic concepts such as deadlocks or liveness are alien to the world of physics, while adherence to physical first principles such as the conservation of energy are seldom captured by computing abstractions. As a consequence, cooperative controllers are often organized varied forms of hierarchical control structures, which split responsibilities by distributing control tasks among well defined layers. This way, each type of controller can be designed by applying the best of breed techniques available to that specific domain. Salient examples are e.g. model checking techniques for the verification of software implementing discrete-event controllers, and state space analysis of dynamic systems to design discrete-time controllers for continuous systems (Gokbayrak and Cassandras 2000). Paradigmatic examples of interest in this work are data collection missions assigned to Unmanned Aerial Vehicles (UAV). In these scenarios a higher layer *supervisory control* deals with the adaptive planning of convenient navigation paths and tasks, while a lower layer *regulation control* deals with driving the forces that maintain position, velocity and orientation of the vehicle (Karimoddini et al. 2014).

Despite the success of this approach, new challenges arise when limited shared resources impose restrictions that compete with each other. In order to reduce UAV costs, weight and size, a single on-board processor can be shared. The same CPU can control both: a) the correct and stable flight of the UAV (in the presence of perturbations, e.g. wind gusts) and b) the reliable collection of data from scattered sources (in the presence of adverse conditions, e.g. fluctuating wireless channels). Depending on unpredictable environmental conditions a dynamic trade off must be applied: maximize the effective throughput of data collection while minimizing the sacrificed stability. In other words, there is a competition for CPU cycles between the supervisory control and the regulation control. This raises the need for the co-design of discrete-event, discrete-time and continuous controllers, i.e. a modular hybrid control design problem.

The aforementioned layered approach calls for sound methodologies to assist the design process, allowing integrative testing of the controllers under a unified platform and in a safe way. In particular for the case of UAVs, reducing testing risks and costs is mandatory. Modeling and simulation-driven engineering has proven a successful strategy to support end-to-end designs of hybrid controllers (Castro et al. 2009). We assess as essential to count on an integrative framework that supports the interconnection of heterogeneous components in a seamless way. In this work we show how the DEVS (Discrete Event Systems Specification) framework (Zeigler et al. 1995, Zeigler et al. 2000) provides an efficient modeling paradigm and simulation mechanism to develop general hybrid systems.

We will present a simulation-based case study for the design, verification and partial validation of a hybrid control system for an UAV. The validation includes flying a real hexacopter subject to controlled maneuvers and a subsequent replication using trace-driven simulations. The vehicle is assigned with the mission of retrieving information from wireless sensors. The UAV's flight robustness and communication efficiency will compete for CPU resources, which will translate into adaptive sampling periods for sensors, controllers and actuators. Then a supervisory control will be requested to enforce the trade-off in a safe way, and update its objectives observing that switching sampling periods could unstabilize the UAV.

2 Motivating Case Study: UAV-based data collection

The combination of UAVs with data-acquisition from sensors is not new. Examples are the monitoring of natural phenomena such as earthquakes or volcanic activity. In (Werner-Allen et al. 2005, Werner-Allen et al. 2006, Song et al. 2009) the collection and study of data attempts to predict the occurrence of events and prevent human and material losses. Also emergency scenarios such as firefighting (Ollero et al. 2007) require different types of data to be exchanged while no communication infrastructure is operative.

Inspired by these applications we consider a scenario where a number of wireless devices are randomly scattered on a field, and we need to collect data stored in them. The exact location of each device is ignored a priori, so we need to explore the terrain and discover nodes during the mission.

When the UAV detects a patch with new sensors, it switches to a hovering state to allow for communication establishment and information retrieval. The UAV will hover until the transmission is completed. When more than one node is detected it is convenient that all available information is retrieved from all sensors within the range of coverage during the same hovering period. We shall consider that a) the same on-board computer that executes the control algorithm which stabilizes the vehicle is also in charge of the communication and b) the computer is a low-cost device of modest capacity.

This creates a compromise between competing tasks: one needs resources to acquire information from nodes, while the other needs to execute the control algorithm often and fast enough to hold the vehicle in a desired position. Wind gusts produce transient responses in the UAV's angle. So a regulation control is placed to counteract the effect of disturbances and keep the desired position of the UAV. We impose that the UAV should observe a certain degree of alignment with the sensors in the land to establish an adequate communication, being the ideal condition when the UAV's angle is zero. When threshold angles are surpassed due to external perturbations, the communication should be interrupted and then restored after the transient response vanished.

2.1 Proposed Approach: controlled trade-off between stability and communication

The approach we present here considers a supervisory controller which must be capable of: deciding how to reduce the sampling-period of the attitude controller as a function of the number of detected sensors to favor data acquisition and interrupting data collection while the UAV is being disturbed.

We subdivide the flight mission into stages according to the tasks (or modes) the UAV must perform. These are: Take-off, Traveling, Hovering and Landing. A mission starts with a Take-off phase. Then, during a Traveling phase the UAV scrutinizes the ground looking for sensors. While in Hovering, the UAV must retrieve data from the sensors detected in the previous step. We define that Traveling and Hovering phases are interleaved. Every mission ends with a Landing phase. The planning for an optimal coverage of the terrain during Traveling phases are out of the scope of this work.

Then, while in Traveling stage, we request a supervisory controller to select the *minimum achievable sampling-period* (imposed by the electronics of the the UAV)for the regulation controller, which will be referred to as the *nominal period* h_0. Using this value the UAV exhibits the maximum control robustness to counteract wind disturbances, with controller using all CPU cycles to update the required forces. Yet, the UAV is only able to execute flight control tasks, and no data-retrieval can be performed.

When sensors are detected, the sampling-period is relaxed ($h > h_0$) in order to assign some CPU time to the communication task. The more relaxed (bigger) the period is, the more available CPU cycles to upload data (which could translate into more sensors surveyed per mission given a fixed battery lifetime). Meanwhile, the sampling-period is also upper-bounded with $h \leq \bar{h}$ by the UAV stability. In the case of a fixed bandwidth $\bar{h}$ will determine the maximum number of sensors that can be simultaneously accessed.

Within the $h_0 < h < \bar{h}$ interval a suitable h must be selected in order to assign the optimal proportion of processor capacity according to the number of detected sensors. There exist guarantees that the vehicle can remain stable under sampling-period switching, based on known results of control theory that provide us with explicit algebraic expressions to adapt the regulation controller.

During Take-off and Landing the regulation controller uses h_0 as no communication is required.

3 BACKGROUND

3.1 Hybrid Systems Modeling and Simulation with DEVS

Modeling and Simulation of hybrid systems is central to understand the behavior of discrete and continuous dynamics interacting with each other. This is because in most cases of practical interest hybrid systems don't accept analytical solutions.

DEVS is a formal model description framework equipped with an abstract simulation algorithm that is independent on the nature of the described system. It has been shown that DEVS can describe exactly any discrete system and approximate continuous systems with any degree of desired accuracy, therefore being capable to simulate all kinds of hybrid systems that undergo a finite amount of changes in finite intervals of time (Zeigler, Praehofer, and Kim 2000).

A system modeled with DEVS is described as a modular and hierarchical composite of submodels, each of them being behavioral (atomic) or structural (coupled). Sumbodels interact by means of events sent through input/output ports. A DEVS Atomic model is defined by the following tuple:

$$A = \{S, X, Y, \delta_{int}, \delta_{ext}, t_A, \lambda\}$$

where S is the set of internal states, X is the set of accepted external events, and Y is the set of available outputs. Four dynamic functions define behavior: $t_A(s) : S \to \mathbb{R}_0^+$ is the lifetime of each state $s \in S$. After $t_A(s)$ units of time an internal state transition $\delta_{int} : S \to S$ is triggered (assuming no external input events arrived). $\delta_{ext}(s, e, x) : S \times X \times \mathbb{R}_0^+ \to S$ is the external state transition function that is triggered when an input event arrives, with $0 \leq e < t_A$ being the elapsed time in a given state. Every time a new state s' is selected with δ_{int} or δext, a new $t_A(s')$ is calculated and the elapsed time e is reset. Finally, $\lambda(s) : S \to Y$ is the output function that can be invoked to send output events only when an internal transition is triggered.

An external transition is triggered every time an input in X is received. This change is performed instantaneously. On the other hand, the output function is executed when t_A has elapsed since last event. Simultaneously, an internal transition is produced.

A DEVS Coupled model interconnects atomic and coupled components together thorough their input/output ports. It can be described by the following tuple:

$$C = \{X, Y, D, EIC, EOC, IC, Select\}$$

where: X and Y are sets of input and output evens respectively, D is the set of components names, IC is the set of internal couplings among members of D, EIC is the external inputs coupling relation (set of couplings between external input ports and internal components) and EOC is the external output coupling relation. Select is a tie-breaking function to assign execution priorities when two or more internal or external transition functions are scheduled for the same simulation time.

The DEVS formalism is closed under coupling (i.e., any hierarchical coupling of DEVS atomic and coupled models defines an equivalent atomic DEVS model).

For practical modeling and simulation we adopted PowerDEVS (Bergero and Kofman 2010), an open-source simulation toolkit based on the DEVS formalism that is particularly oriented to hybrid systems.

DEVS Graph (Zeigler et al. 1995) is a graph-like language to describe the behavior of DEVS atomic models in a visually intuitive way. Each state $s \in S$ is indicated by a bubble with its name and lifetime, and arcs connecting bubbles denote state transitions. Dashed lines indicate internal transitions and solid lines indicate external transitions. The condition that must be satisfied by an input signal *In* to trigger an external transition when carrying a value *val* is indicated as: `In?val`. An output can only be produced after a state the lifetime has elapsed. An output of a *val* value on an *Out* port is indicated as: `Out!val`. We will use this notation to depict discrete-event behavior in Section 6.1 for supervisory controllers. The implementation of a DEVS model in PowerDEVS departing from a DEVS Graph diagram is straightforward.

3.2 Stability Analysis

Let $x(t) \in \mathbb{R}^n$ be a state vector whose evolution is described by a linear time-invariant system with input $u(t) \in \mathbb{R}^m$, more precisely:

$$\dot{x}(t) = A\,x(t) + B\,u(t), \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are the state and input matrices, respectively. If we sample $x(t)$ by taking values at $t = kh$ for $k \in \mathbb{N}$, where $h > 0$ is the sampling-period and assuming that $u(t)$ is constant between samples, then equation (1) can be discretized as follows:

$$x(k+1) = \Phi(h)\,x(k) + \Gamma(h)u(k) \tag{2}$$

with $x(k) = x(hk)$, $\Phi(h) = \exp(Ah) \in \mathbb{R}^{n\times n}$ and $\Gamma(h) = \int_0^h \exp(A\tau)\,d\tau B$, where $\exp : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is the matrix exponential. To control the system behavior we use state-feedback, that is $u(k) = K(h)x(k)$, where $K(h) \in \mathbb{R}^{m\times n}$ is the state-feedback gain which depends on the sampling-period h. So closed loop equation is given by:

$$x(k+1) = \Big(\Phi(h) + \Gamma(h)K(h)\Big)x(k) \tag{3}$$

The behavior of the state is determined by the closed-loop matrix $\Phi^{CL}(h) = \Phi(h) + \Gamma(h)K(h)$. Considering a finite set of admissible sampling-periods $0 < h_0 < \cdots < h_N$, we have a family $\Sigma = \left\{\Phi_i^{CL}\right\}_{i=1,\ldots,N}$ of subsystems, where $= \Phi_i^{CL} = \Phi^{CL}(h_i)$, $i = 0, ..., N$. Thus, the evolution of $x(k)$ is a function of the active subsystem in Σ, which is time-dependent. Then, equation (3) can be seen as a linear autonomous Discrete-Time Switched System (DTSS) (Liberzon 2012):

$$x(k+1) = \Phi_{\sigma(k)}^{CL}x(k) \tag{4}$$

where $\sigma : \mathbb{N} \to \{0,\ldots,N\}$ is a piecewise constant function, which defines the switching sequence. In our case study the switching function σ is defined according to whether the UAV is retrieving data and the number of sensors. It is mandatory to guarantee the stability of the overall system for any possible switching sequence. In a switching system the stability of each individual subsystem is not a sufficient condition. As it was stated in (Narenda and Balakrishnan 1994) the commutativity between every pair of matrices in Σ ensures that a common quadratic Lyapunov function (CQLF) exists, and then the stability of the switching system is guaranteed. In (Felicioni and Junco 2008, Felicioni et al. 2010) the authors use this result to design adaptive control laws that guarantee stability for any arbitrary switching sequence $\sigma(k)$. We shall rely on this result to adapt sampling periods dynamically with stability guarantees.

4 CONTINUOUS UAV MODEL AND DISCRETE REGULATION CONTROLLER

An UAV can be described as a rigid-body with six degrees of freedom $(6-DOF)$. In particular for a multi-rotor vehicle, the position and its orientation are coupled. More specifically, to modify the lateral position of the vehicle it is necessary to modify its attitude. In fact, the attitude of the multi-rotor is the main variable to control, once the attitude is stabilized it is possible to control the vehicle position and eventually reject disturbances.

The state vector $x(t) \in \mathbb{R}^6$ specifies the attitude of the vehicle: $x = \left[\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}\right]^T$ as seen in Figure 1, where ϕ, θ and ψ are the pitch, roll and yaw angles, respectively. This rigid-body is subject to forces and moments: $u = \left[F_x, F_y, F_z, \tau_\phi, \tau_\theta, \tau_\psi\right]^T$. The dynamic equations which describe its movement are obtained by considering Euler-Newton equations (5)-(7) where $\mathbf{F}$ is the vector of forces (control input), $\mathbf{M}$ the moments vector (control input), $\mathbf{V}$ is the velocity of the vehicle, ω the angular velocity, m is the mass of the UAV and I is the inertia matrix. These quantities are expressed in the (non-inertial) vehicle frame. However, given the attitude of the vehicle (ϕ, θ, ψ) it is possible to refer the velocity of the UAV and its position in an inertial frame.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin(\phi)tg(\theta) & cos(\phi)tg(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & \frac{sin(\phi)}{cos(\theta)} & \frac{cos(\phi)}{cos(\theta)} \end{bmatrix} \omega \tag{5}$$

$$m\dot{\mathbf{V}} = -\omega \times \mathbf{V} + \mathbf{F} \tag{6}$$

$$I\dot{\omega} = -\omega \times (I\omega) + \mathbf{M} \tag{7}$$

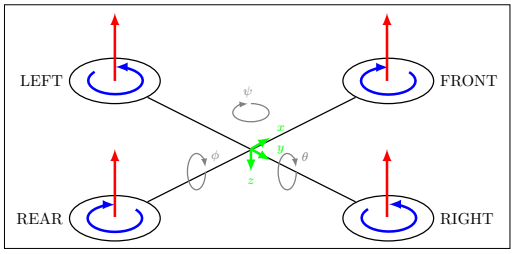


Figure 1: Quadrotor attitude.

This is a simplified model that does not take into account the behavior of actuators and sensors, but is good enough for the application studied in this work. For small angles near hovering, i.e. $\phi \simeq 0$, $\theta \simeq 0$ and $\psi \simeq 0$, the non-linear model given in equations (5)-(7) can be linearized. Moreover, the multi-rotor attitude dynamical model can be decomposed into three decoupled dynamical subsystems, one for each attitude angle, see Figure 1. For each of these angles we have the following model:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ b_0 \end{bmatrix} \tau(t)$$

where $b_0 = 1/I$ corresponds to the inverse of the inertia and $\tau(t) \in \mathbb{R}$ is the torque in the corresponding axis. State $x(t) \in \mathbb{R}^2$, represents the angle and angular velocity in the corresponding axis, for instance for roll angle, $x = [\phi, \dot{\phi}]$. Following the ideas of Subsection 3.2, this model can be discretized. For each subsystem, a state-feedback controller is designed, such that: $\tau(k) = K(h_i)x(k)$, with $K(h_i) = \begin{bmatrix} K_1(h_i) & K_2(h_i) \end{bmatrix} \in \mathbb{R}^{1\times 2}$.

Therefore, discrete-time closed-loop dynamics for each sampling-period $h_i = 0,...,N$ is given by:

$$x(k+1) = \left(\begin{bmatrix} 1 & h_i \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{b_0 h_i^2}{2} \\ b_0 h_i \end{bmatrix} K(h_i) \right) x(k) = \Phi_i^{CL} x(k) \tag{8}$$

Considering the different flight modes, the family of closed-loop matrices $\Sigma = \{\Phi_0^{CL}, \ldots, \Phi_N^{CL}\}$ should be stabilized for arbitrary switching. During a mission, a UAV acts as a data-mule and as it moves forward in its path it will be establishing communication with a different number of ground sensor nodes.

As it was mentioned in Section 3.2, ensuring stability for the system under arbitrary switching sequence is equivalent to asking for closed-loop matrices in Σ to be pairwise commutative. This approach can be easily applied by means of the algebraic equations:

$$K_1(h_i) = -4K_1(h_0)/\xi \quad , \quad K_2(h_i) = 2(h_0 K_1(h_0) - n h_0 K_1(h_0) - 2K_2(h_0))/\xi \tag{9}$$

with $n = h_i/h_0$ and $\xi = -4 + (n-1)b_0 h_0 (n h_0 K1(h_0) + 2K_2(h_0))$. Notice that, in order to calculate $K(h_i)$ for every $i = 1,...,N$, we require to known $K(h_0)$. Of course, $K(h_0)$ should be such that $\Phi^{CL}(h_0)$ is stable.

In Figure 2 we can see how the adaptive discrete-time controller is connected to the UAV. Figure 3 shows our implementation of the system in PowerDEVS, using the adaptation law given in eq. (9).

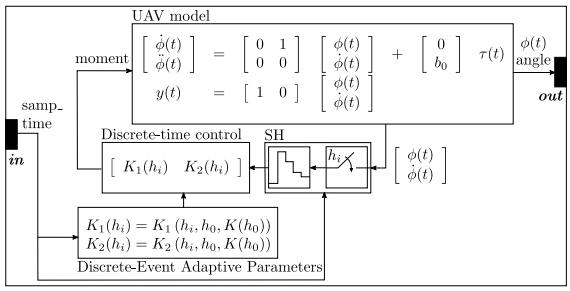


Figure 2: Schematic of the continuous UAV system and the adaptive discrete-time/event controller.

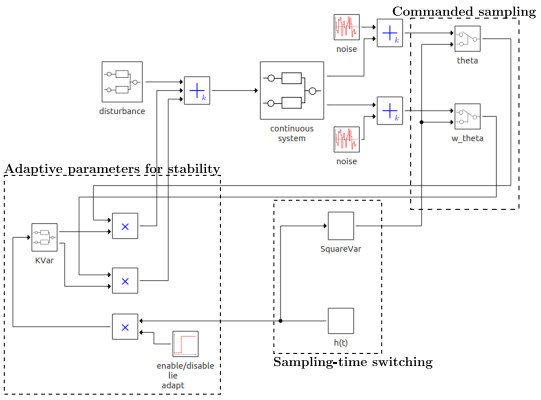


Figure 3: System model in PowerDEVS.

We will stress on the necessity of adapting the vehicle control law in a context of switching sampling-period with the following simulated example. Let $h_0 = 5$ ms, $h_1 = 40$ ms, $h_2 = 290$ ms and the switching sequence σ followed by sampling-period: h_0 over h_0, then h_1 for h_1 and finally h_2 during h_2. The sequence is repeated over time. If we do not conduct any adaptation, i.e. retain controller gains for h_0 called nominal gains hereafter, the result is an unbounded growth of the state values, i.e. an unstable system (see Figure 4). The matrix $\Phi^{CL}(h_0)\Phi^{CL}(h_1)\Phi^{CL}(h_2)$ has an unstable eigenvalue. Yet, each sampled subsystem is stable.

5 VALIDATION EXPERIMENTS

5.1 Experiments Performed with the UAV

In this section, we analyze a test carried out with an hexacopter (see Figure 5) emulating a flight mission. This vehicle uses an autopilot called ChoriCopter. For this experiment we defined $h_0 = 5$ ms, $h_1 = 15$ ms and $h_2 = 25$ ms. In Figure 6 we can see the switching sequence followed by the sampling-period driving the regulation controller. The UAV starts in Take-off mode and then switches to Traveling mode (both with h_0) following a predefined path to collect data. At 26 s the UAV detects a group of N_2 sensors. Thus, it evolves

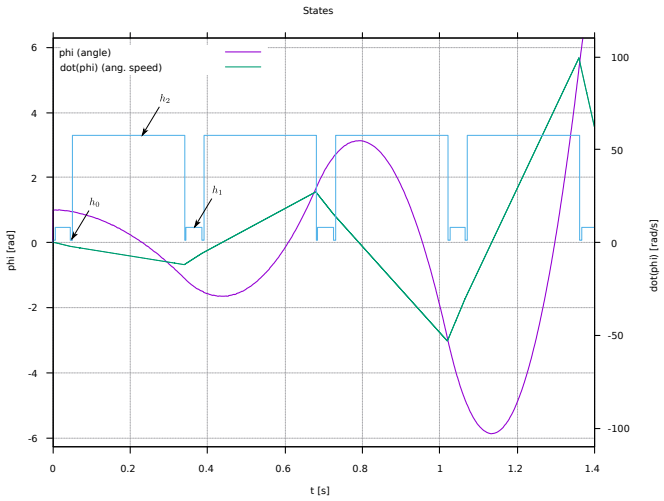


Figure 4: Continuous-state unstable behavior under sample-time switching.

to Hovering mode and the supervisory controller assigns h_2 to the controller. Once data-retrieval is finished after about 15 s the UAV returns to Traveling and the supervisory controller switches the sampling-period back to h_0. Around 58 s the vehicle detects a group of N_1 sensors and then jumps to Hovering again. In accordance with this value, the supervisory assigns h_1. The cycle of Traveling and Hovering modes with different sampling-period is repeated.

As it can be noticed from Figure 6, switching signal has some spurious glitches. This is due to some packets missed in the communication between the UAV and the computer used to monitor the experiment.

The measured *roll angle* can be seen in Figure 7a. While in Hovering the angle reference is null whereas the current angle is not. This bias is could be attributed to the lack of integral action in the regulation controller. The glitches around 22 s are related to the transition between Take-off and Traveling modes.

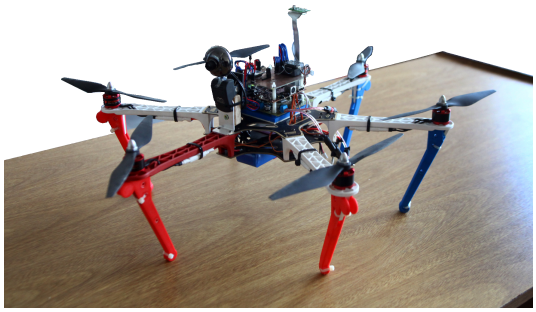


Figure 5: Hexacopter used in the experimental tests.

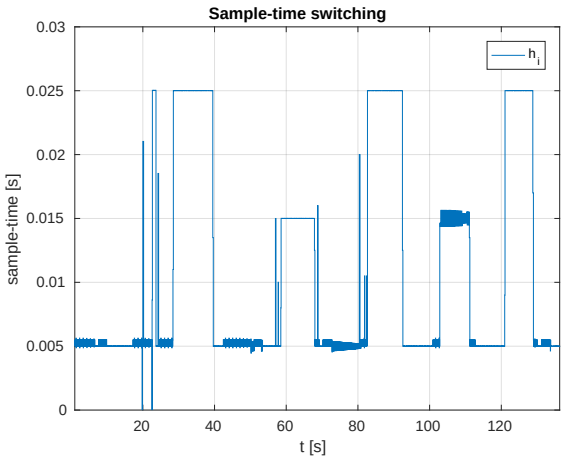
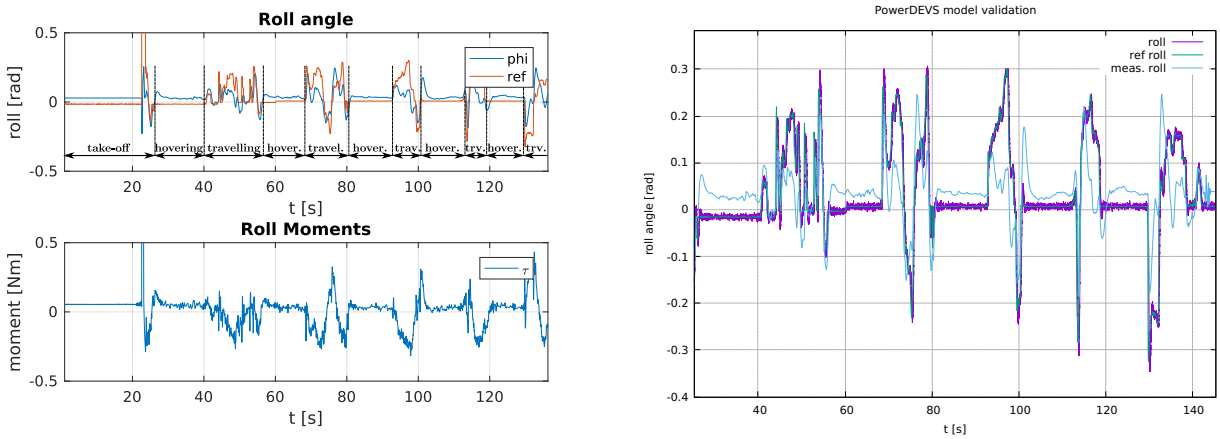


Figure 6: Manually-imposed sampling-period.

5.2 Trace-Driven Simulation Validation

We used experimental data recorded during the tests in the previous section to validate our model implemented in PowerDEVS. We took the angle reference and sampling-period signal traces to drive the simulation model. Figure 7b illustrates the resulting curves. Measured and simulated data are overlapped to check

the degree of accuracy. The simulated angle follows the measured reference closely. However, measured and simulated roll angles differ in an offset. Even without capturing this effect in our model, we conclude that the approximation is quite acceptable for our purposes.



(a) Measured roll angle and moment.

(b) Roll angle comparison between 25.5 s and 145.2 s.

Figure 7: Evolution of measured and simulated $\theta(t)$ during an experiment.

6 SUPERVISORY CONTROL DESIGN: DISTURBANCE REJECTION VS. COMMUNICATION EFFICIENCY

The strategy followed here to implement the trade-off mentioned in Section 2 increasing the sampling-period of the vehicle controller, when needed, so as to have more time to acquire information. The amount of time necessary to execute the control task T_{ctrl} is determined by the CPU and is a fixed value. The sampling-period used during the Traveling stage, when no data is being collected, equals this value. A sampling period is divided into one part when the CPU is devoted to execute control task and the remainder where it is used to collect data. The ratio between these two times is the data communication efficiency η_{data}. If we limit the time while the UAV is collecting data not to waste energy, we will need to enlarge the sampling period so as to collect a bigger amount of data. The previous trade-off is then expressed as the relation between the sampling-period and its part devoted to data acquisition.

We discussed already that we can't use always the minimum value for sampling-period T_{ctrl} because with this value no data can be collected. But, why don't we always use the maximum value, $\bar{h}_i$? The reason is that disturbance rejection in this case is the worst that can be achieved.

The UAV is required to remain hovering collecting data for a time $\bar{T}$, no matter how many sensors the UAV can communicate to. This time limit is set in order to rationalize energy consumption, and is valid only in the absence of disturbances. Otherwise it will take longer than $\bar{T}$ to collect a same amount of data.

Once a communication protocol between the UAV and the sensors is defined, the bandwidth BW gets fixed. Thus, the time needed to collect all data by a group of N sensors is given by (10), where L_{msg} is the message length and $T_{tx} < \bar{T}$. During this time the CPU is exclusively devoted to data acquisition, and the remainder $\bar{T} - T_{tx}$ is used for the vehicle controller execution. The number of sampling periods that fit within $\bar{T}$ is $n_{msg} \in \mathbb{N}_0$, and it is calculated as in (11). This is the number of executions of the vehicle controller conducted while performing the overall data transmission.

$$T_{tx} = \frac{NL_{msg}}{BW} \qquad (10) \qquad n_{msg} = \frac{\bar{T} - T_{tx}}{T_{ctrl}} \qquad (11) \qquad h_i = T_{ctrl} + \frac{T_{tx}}{n_{msg}} \qquad (12)$$

The sampling-period for the regulation controller is obtained from the expression in (12). This value represents the minimum sampling-period needed to collect all data in time $\bar{T}$. If the sampling-period were longer than this all data could be gathered in a shorter time but at the expense of being less robust to disturbances. Finally, the portion of a single h_i devoted to data acquisition is given by $T_{meas} = h_i - T_{ctrl}$.

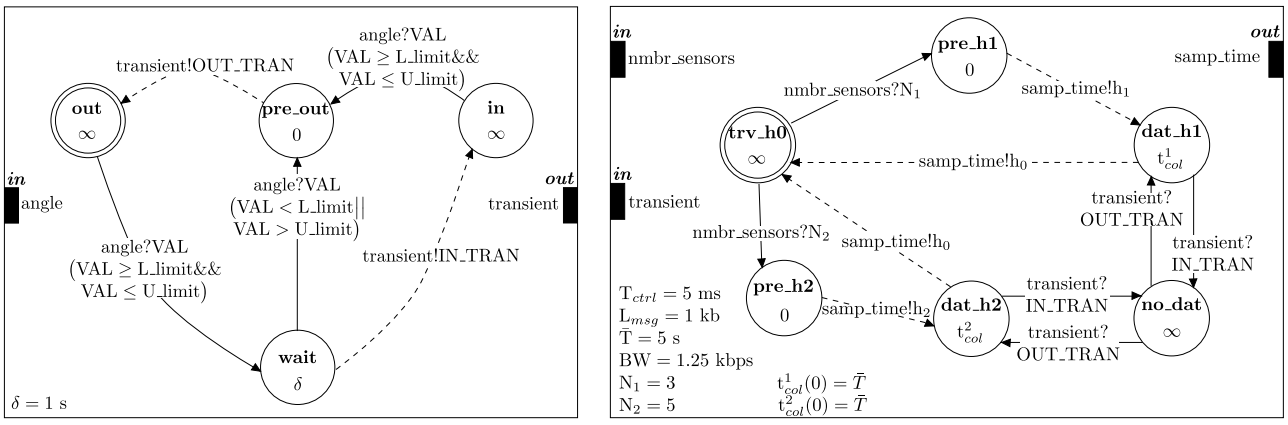
As it was already mentioned, the data communication efficiency is the ratio between the fraction of the sampling period used for data collection and the sampling-period itself. But on the other hand, it is also the quotient between the time during which the UAV is required to be in hovering and T_{tx}: $\eta_{data} = T_{meas}/h_i = \bar{T}/t_{tx}$. Its value is bounded by $0 \leq \eta_{data} \leq \bar{\eta}_{data} < 1$, since the sampling-period is limited by $T_{ctrl} \leq h_i \leq \bar{h}_i$. Finally, the expression in (13) relates sampling-period with the number of sensors N. Since h_i is upper bounded by $\bar{h}_i$, then this value determines $\bar{N}$. Let $\xi = \bar{T}BW/L_{msg}$, then

$$h_i = \begin{cases} T_{ctrl}\xi/(\xi - N) & \text{if } 0 \leq N \leq \bar{N} \\ T_{ctrl}\xi/(\xi - \bar{N}) & \text{otherwise} \end{cases} \tag{13}$$

6.1 DEVS Model Specification

The DEVS Graph corresponding to the Transient Detector is shown in Figure 8a. This block acts as an interface between continuous and discrete states. The input of this detector is the UAV angle, whereas its output is a binary signal indicating whether the UAV is undergoing a transient response. This output feeds the Supervisory Controller. The Transient Detector starts in the **out** state, indicating that deviation from hovering is outside preestablished bounds set by L_limit and U_limit parameters, respectively. The state remains in **out** until a new angle information lies within bounds. In that case, an external transition is triggered and the state jumps to **wait**. There, it waits for a predefined stabilization time δ after which an internal transition is triggered, changing its state to **in** and outputting an IN_TRAN signal. If while in the **wait** state the angle input falls outside bounds, an external transition would jump back to **out** state but producing an OUT_TRAN output. Once in the **in** state, it remains there while the angle is inbound. Otherwise, an internal transition would switch state to **out** and produce an OUT_TRAN output.

This controller was implemented as a DEVS coupled model. It consists of an in/out band detector, made of two comparators, and the Transient Detector itself which is a DEVS atomic model.



(a) Transient response detector

(b) Supervisory controller

Figure 8: Communication and processing resources allocator unit.

Figure 8b exhibits the DEVS Graph representing the Supervisory Controller. It is defined by a DEVS atomic model with two inputs and one output. First input corresponds to the signal provided by the Transient Detector to decide weather angle conditions are suitable to collect data. Through the other input the controller receives the number of detected sensors. For the present case, this can take values: 0, N_1 or

N_2. Every time a group of sensors is detected, an input to this atomic model is produced. The supervisory controller then decides which sampling-period should be used: h_0, h_1 or h_2 and outputs this value through its corresponding output port.

The Supervisory Controller starts in **trv_h0** state (Traveling phase defined in Section 2.1). Depending on the number of sensors detected, N_1 or N_2, the state jumps to **dat_h1** or **dat_h2** respectively. Before reaching these states, the corresponding sampling-period computed with (13) is outputted. The Supervisory Controller will remain in either **dat_h1** or **dat_h2** while the UAV is collecting data, for t_{col}^1 or t_{col}^2 units of time. In absence of disturbances, these variables equal $\bar{T}$. Otherwise, if a disturbance is experienced strong enough to take the angle out of bounds, an external transition to **no_dat** state is produced. When the Transient Detector indicates that deviation from hovering is considerable then data acquisition is stopped. After resuming from **no_dat**, the acquisition in either **dat_h1** or **dat_h2** continues for the difference between $\bar{T}$ and the elapsed time before the transient took place. In this case, the time needed to collect all stored data gets longer. Once all the information is acquired, an internal transition is produced and the supervisory state jumps back to **trv_h0**, with sampling-period h_0.

The Supervisory Controller in Fig. 8b is formally defined as a DEVS atomic model $Superv\_Ctrl = \{S, X, Y, \delta_{int}, \delta_{ext}, t_a, \lambda\}$ whose pseudocode can be seen in Table 1.

```
S: {trv_h0, pre_h1, pre_h2, dat_h1, dat_h2, no_dat, sigma}
X:  {(nmbr_sensors, {N1, N2});
     (transient, {IN_TRAN, OUT_TRAN})}
Y:  {(samp_time, {h0, h1, h2})}
δint(s): if (s == pre_h1){ // Internal Transition
             s = dat_h1;
             h = h1;
             sigma = Tbar;}
         else if (s == pre_h2){
             s = dat_h2;
             h = h2;
             sigma = Tbar;}
         else if (s == dat_h1) {
             s = trv_h0;
             h = h0;
             sigma = INF;}
         else if (s == dat_h2) {
             s = trv_h0;
             h = h0;
             sigma = INF;}
δext(s,e,x): if (x.port == 0){ // External Transition
             if ((s == trv_h0)&&(*xv == N1)){
                 s = pre_h1;
                 sigma = 0;}
             if ((s == trv_h0)&&(*xv == N2)){
                 s = pre_h2;
                 sigma = 0;}}
             if (x.port == 1){
                 if ((s == dat_h1)&&(*xv == 0)){
                     elapsed_h = e;
                     s = no_dat;
                     sigma = INF;}
                 if ((s == dat_h2)&&(*xv == 0)){
                     elapsed_h = e;
                     s = no_dat;
                     sigma = INF;}
                 if ((s == no_dat)&&(*xv == 1)){
                     if (h == h1) {
                         s = dat_h1;
                         sigma = Tbar - elapsed_h;}
                     if (h == h2){
                         s = dat_h2;
                         sigma = Tbar - elapsed_h;}}}
λ(s):    if (s == pre_h1){ // Output
             y[0] = h1;}
         else if (s == pre_h2){
             y[0] = h2;}
         else if ((s == dat_h1) || (s == dat_h2)){
             y[0] = h0;}
         else if (s == trv_h0){
             y[0] = h0;
             sigma = INF;}
         return Event(&y,0)
ta(s):   return sigma // Time advance
```

Table 1: Pseudocode for the *Superv_Ctrl* DEVS Atomic Model.

Figure 9 illustrates the resulting overall hybrid DEVS model structure. The **Supervisory Controller Unit for Resources Allocation** at the top has an input to receive the number of detected sensors. The output port of this block is connected to the **Continuous Dynamic** block at the bottom. Through this connection the current sampling-period is sent to the adaptive discrete-time regulation controller. This is also an event-based signal. The output from the Regulation Controller is the continuous moment applied

to the UAV by the actuators. The tilt angle, which is also a continuous signal, is connected to the Transient Detector. Finally, the transient detector output is also an event-based signal.

7 SIMULATION RESULTS

We performed tests on the model of Figure 9 considering the values shown in Table 2. The data collection efficiency is bounded by $0 \leq \eta_{data} \leq 0.8$. Figure 10 illustrates the sampling-period in (13) for $N \in [0, 10]$. With these parameters, groups of sensors larger than 5 saturate the sampling-period.

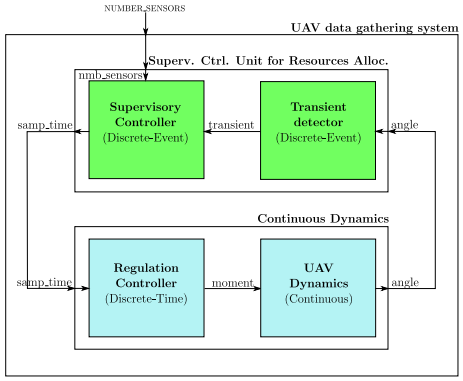


Figure 9: Continuous Dynamic and Supervisory Control Units for Resource Allocation with Stability Constraints.

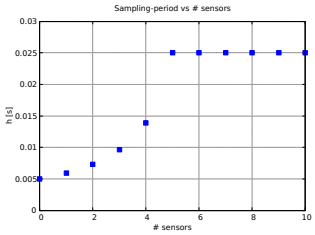


Figure 10: sampling-period vs. Number of sensors.

Parameter	Value
BW	1.25 kbps
L_{msg}	$1kb$
$\bar{T}$	5 s
T_{ctrl}	5 ms
$\bar{h}_i$	25 ms
N_1	3
N_2	5
U_limit	0.2 rad
L_limit	-0.2 rad
δ	1 s

Table 2: Parameter values used for testing.

7.1 Verification in Absence of Disturbances

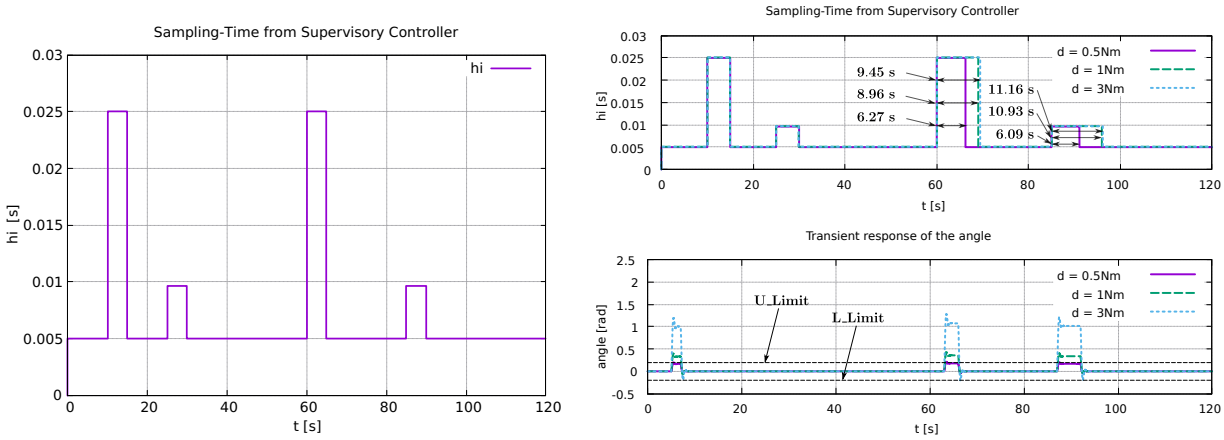
Figure 11a illustrates the results of this simulation. The UAV starts Traveling. At $t = 10$ s it detects a group of N_2 sensors. It takes $\bar{T}$ to collect all data from them in Hovering stage. After finishing, the UAV switches to Traveling again. At $t = 25$ s, the vehicle detects N_1 sensors. It switches to Hovering and takes $\bar{T}$ again to collect all data. After that, it travels between 32.5 s and 60 s. At $t = 60$ s and 85 s, N_2 and N_1 are detected respectively. Each switching of sampling-period is followed by the execution of the adaptation algorithm (9) for the regulation controller.

7.2 Verification by Considering Disturbances

Figure 11b shows the same experiment but now carried out in the presence of disturbances. These are represented by momentum pulses which emulate a sudden wind gust. We simulate the influence of three different disturbance durantions and amplitudes.

Disturbances appear at 5 s, 63 s and 87 s (and last until 7 s, 66 s and 92 s) while the vehicle regulation controller is using h_0, h_2 and h_1 respectively. That means that the first and second data acquisition (Hovering) phases (between 10 s and 15 s and between 25 s and 30 s) are performed without being altered by wind. Disturbances cause a transient response in the angle of the UAV. Depending on the intensity of the gusts, the angle response can escape from the stability band limited by U_limit and L_limit, making the supervisory controller to stop data acquisition. These limits are shown in Figure 11b with dotted lines. Once the angle returns within the band for a time longer than δ the supervisor decides to resume data acquisition. The effects of disturbances in the third and fourth acquisitions are clearly seen in Figure 11b (upper pane) where the width of the hovering periods vary according to the strength and length of each disturbance.

On the one hand, it can be concluded that the bigger and longer the disturbance the longer the time used to collect all data. On the other hand, the higher the sampling-period, the less robust to a same disturbance the system becomes, which imply bigger transient response amplitudes and longer settling times.



(a) Without disturbances

(b) Considering disturbances

Figure 11: sampling-period during acquisition.

This highlights the necessity to relax the vehicle controller sampling-period only when necessary and no more than strictly required.

8 CONCLUSIONS AND FUTURE WORK

We presented a modeling and simulation-assisted design of a hybrid supervisory controller for an unmanned aerial vehicle (UAV) used for data collection from randomly distributed wireless sensors. In our study the UAV requires to adapt scarce processor capacity allocation under competing cyber-physical constraints: stability vs. data communication. A supervisory control layer resorts to sampling-period relaxation of the vehicle regulation control layer. To guarantee stability under switching of sampling-periods, we consider a control law adaptation that allows to calculate and update safe control gains.

From a methodological point of view, the study required to reason about the problem at three different system levels or domains: continuous, discrete-time and discrete-event. The continuous dynamics of the UAV physics are modelled with differential equations. A discrete-time proportional-derivative controller is implemented so as to fly the UAV and stabilize it while hovering in the presence of disturbances. Two kinds of discrete events are considered, which are driven by unpredictable dynamics: the sudden detection of a new number of sensors and the intervention of wind gusts.

We proposed a control mechanism to trade-off processor resources between stability and communication constraints. We also established bounds in the angle of the UAV to consider it in a acceptable position to enable the communication. The combinations of these dynamics yield unpredictable sequences of discrete events. The strategy based on adapting the sampling-period of the regulation controller created the need for guaranteeing stability under period switching. Thus, depending on discrete-event dynamics, we adapted the discrete-time dynamics that control the stability of the continuous dynamics.

We successfully followed a modeling and simulation-driven engineering approach relying on the DEVS framework to support the composition of hybrid domains in a seamless way. We achieved model continuity, designing the system iteratively and incrementally. We smoothly used the same DEVS models and simulator to a) verify the hypothesis of potential destabilization of the continuous system under certain sampling-period switching sequences , b) validate the continuous model under switching conditions by flying a real hexacopter, recording responses and driving simulations with the recorded traces, c) design an event-based control strategy to adapt the discrete-time controller with stability guarantees, and d) verify some scenarios where varying number of sensors and wind conditions create competition for processor resources.

As our design is a proof-of-concept there is room for several improvements to consider as future work. One option consists of adding a time limit to be used when the supervisory control is in the **no_dat** state

so that, if reached, this controller would choose a lower sampling-period and thus reduce the number of sensors to be read simultaneously. This action would improve disturbance rejection on the one hand, but on the other hand would increase the time needed to collect all the data from the sensors. Hence, the supervisory controller should select a subgroup of sensors from the group. The improved aim would be not to waste time trying to collect all data from a group of sensors since they might have some degree of spatial correlation. Another important variable to take into account is remaining battery charge, which could be included in the decisions of the supervisory controller. finally, as the number of states of the supervisory controller increases, it is worth exploring automatic controller synthesis techniques that can guarantee desirable features such as deadlock-free operation or liveness properties.

REFERENCES

Bergero, F., and E. Kofman. 2010. "PowerDEVS: a tool for hybrid system modeling and real-time simulation". *SIMULATION* 87 (1-2): 113–132.

Castro, R., E. Kofman, and G. Wainer. 2009. "A DEVS–based End-to-end Methodology for Hybrid Control of Embedded Networking Systems". *IFAC Proceedings Volumes* 42 (17): 74–79.

Felicioni, F., C. Berbra, S. Gentil, and S. Lesecq. 2010. "On-line adaptive control of a quadrotor under (m,k)-firm constraint". In *XXII Congreso Argentino de Control Automático AADECA 2010, Buenos Aires, Argentina*. AADECA.

Felicioni, F. E., and S. J. Junco. 2008. "A Lie algebraic approach to design of stable feedback control systems with varying sampling rate". *IFAC Proceedings Volumes* 41 (2): 4881–4886.

Gokbayrak, K., and C. G. Cassandras. 2000. "Hybrid Controllers for Hierarchically Decomposed Systems". In *Hybrid Systems: Computation and Control*, 117–129. Springer Berlin Heidelberg.

Karimoddini, A., H. Lin, B. M. Chen, and T. H. Lee. 2014. "Hierarchical hybrid modelling and control of an unmanned helicopter". *International Journal of Control* 87 (9): 1779–1793.

Liberzon, D. 2012. *Switching in Systems and Control*. Systems & Control: Foundations & Applications. Birkhäuser Boston.

Narenda, K. S., and J. Balakrishnan. 1994. "A common Lyapunov function for stable LTI systems with commuting A-matrices". *IEEE Transactions on automatic control* 39 (12): 2469–2471.

Ollero, A., P. J. Marron, M. Bernard, J. Lepley, M. la Civita, E. de Andres, and L. van Hoesel. 2007. "AWARE: Platform for Autonomous self-deploying and operation of Wireless sensor-actuator networks cooperating with unmanned AeRial vehiclEs". In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, 1–6. IEEE.

Song, W.-Z., R. Huang, M. Xu, A. Ma, B. Shirazi, and R. LaHusen. 2009. "Air-dropped sensor network for real-time high-fidelity volcano monitoring". In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, 305–318. ACM.

Werner-Allen, G., J. Johnson, M. Ruiz, J. Lees, and M. Welsh. 2005. "Monitoring volcanic eruptions with a wireless sensor network". In *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, 108–120. IEEE.

Werner-Allen, G., K. Lorincz, J. Johnson, J. Lees, and M. Welsh. 2006. "Fidelity and yield in a volcano monitoring sensor network". In *Proceedings of the 7th symposium on Operating systems design and implementation*, 381–396. USENIX Association.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. 2 ed. Academic press.

Zeigler, B. P., H. S. Song, T. G. Kim, and H. Praehofer. 1995. *DEVS framework for modelling, simulation, analysis, and design of hybrid systems*, 529–551. Berlin, Heidelberg: Springer Berlin Heidelberg.