Demo: Co-simulation of UAVs with INTO-CPS and PVSio-web[*]

Maurizio Palmieri[21] Cinzia Bernardeschi[1]
Andrea Domenici[1] Adriano Fagiolini[3]
[1]Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy

[2]Dipartimento di Ingegneria dell'Informazione, University of Florence, Italy

[3]Dipartimento di Energia, Ingegneria dell'Informazione e Modelli Matematici (DEIM), University of Palermo, Italy

March 28, 2019

Abstract

This demo shows our ongoing work on the co-simulation of co-operative Unmanned Aerial Vehicles (UAVs). The work is based on the INTO-CPS co-simulation engine, which adopts the widely accepted Functional Mockup Interface (FMI) standard for co-simulation, and the PVSioweb prototyping tool, that extends a system simulator based on the PVS logic language with a web-based graphical interface. Simple scenarios of Quadcopters with assigned different tasks, such as rendez-vous and space coverage, are shown. We assumed a linearized dynamic model for Quadcopters formalized in OpenModelica, and a linearized set of equations for the flight control module written in C language. The co-ordination algorithm is modeled in PVS, while PVSio-web is used for graphical rendering of the co-simulation.

1 Introduction

Nowadays, the deployment of multi-UAVs systems is rapidly increasing in many different applications, ranging from precision farming to surveillance, search and rescue, etc. (e.g. [1], [4]). Given the recent introduction of a new co-simulation

[*]Postprint. Published in: Mazzara M., Ober I., Salaün G. (eds) Software Technologies: Applications and Foundations. STAF 2018. Lecture Notes in Computer Science, vol 11176. Springer, Cham. The final authenticated publication is available online at `https://doi.org/10.1007/978-3-030-04771-9_5`

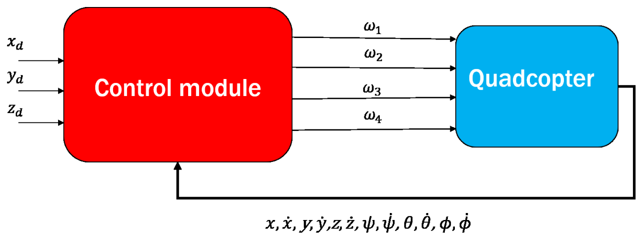

x_d
y_d
z_d
Control module
ω_1
ω_2
ω_3
ω_4
Quadcopter
$x, \dot{x}, y, \dot{y}, z, \dot{z}, \psi, \dot{\psi}, \theta, \dot{\theta}, \phi, \dot{\phi}$

Figure 1: Black box schema of a quadcopter

standard, the Functional Mock-up Interface [2], and tool-kits to exploit such a standard, such as INTO-CPS [5], we combined these technologies with tools for formal modeling, such as PVS [8]. The result is a modular and flexible framework that can be used to co-simulate UAV coordination algorithms dealing with the heterogeneous nature of different UAV models. In this work, we will show an example where the base elements of the FMI co-simulation, the FMUs (Functional Mock-up Unit), are built using different tools (OpenModelica [3], PVSio-web [6], and C code).

In the rest of this section, we provide basic background knowledge of quadcopter representation and consensus algorithm used in the subsequent sections.

1.1 Background on Quadcopters

A quadrotor aircraft, or quadcopter, schematically consists in a cross-shaped chassis supporting one rotor at the end of each arm. The quadcopter's movements are determined by the resultant thrusts and torques of the rotors, which in turn depend on their angular speeds $\omega_1, \omega_2, \omega_3, \omega_4$. The state of the quadcopter is composed of 12 variables: (i) actual position (x, y, z); (ii) linear speeds $(\dot{x}, \dot{y}, \dot{z})$; (iii) attitude, given by the 3 angles pitch, roll, and yaw (ϕ, θ, ψ respectively); (iv) attitude angular speeds $(\dot{\phi}, \dot{\theta}, \dot{\psi})$. The values of $\omega_1, \omega_2, \omega_3, \omega_4$ are computed by the flight control module, which takes as input the desired target (x_d, y_d, z_d) and the actual state of the drone (actual position , linear speeds, attitude and attitude angular speeds) and produces the angular speeds of the four rotors required to reach the target. A simple black box schema of a quadcopter is shown in Figure 1.

1.2 Background on the Consensus Algorithm

We have studied a well-known algorithm proposed in [7] for accomplishing the task of rendez-vous, gathering the drones in a position given by the average of their initial ones. The consensus algorithm can be expressed with the following equation:

$$x_d^{k+1}(i) = x_d^k(i) + \epsilon \sum_{j \in N_i} (x_d^k(j) - x_d^k(i)) \tag{1}$$

where $x_d^k(i)$ is the target position of the drone i at step k, $\epsilon \in (0,1)$ is a parameter of the algorithm, and N_i is the set of neighbors of drone i.

We will consider the case in which N_i is only composed of the preceding and following drones, reducing (1) to

$$x_d^{k+1}(i) = x_d^k(i) + \epsilon(x_d^k(i-1) - x_d^k(i)) + \epsilon(x_d^k(i+1) - x_d^k(i)). \tag{2}$$

2 Co-simulation environment

In this section, we will provide details on the implementation of the FMI co-simulation used to validate the co-ordination algorithm. We have modeled a system composed of many quadcopters, each represented by 3 different subsystems: (i) the physical part of the quadcopter; (ii) the flight control module; (iii) the coordination algorithm.

The physical part of the quadcopter has been represented with a system of linear differential equations for computing the acceleration, the speed, the position and the attitude of the quadcopter based on the angular speed of the four rotors. The system of equations has been written with OpenModelica, which allows us the automatic generation of the FMU.

The flight control module implements a system of linear equations that compute the angular speed of the rotors needed to move the quadcopter toward a target point. The system of equations has been written in C language and embedded in an FMU.

The coordination algorithm has been modeled in the PVS formal language. The PVSio-web toolkit provides the simulation environment for the co-ordination algorithm and the graphical animation of the interface. The PVS model, along with the whole PVS package has been automatically embedded in an FMU using the approach proposed in [9]. The communications between quadcopters are completely abstracted by connecting the output of the coordination algorithm FMU not only with the flight control FMU of the same drone but also with the coordination algorithm FMU of the other drones.

We have created a scenario to test our system with 5 drones and a fixed co-simulation step-size of 0.05 seconds. The parameters of the scenario are shown in Table 1 (Rendez-vous). More precisely, Figures 2a and 2b, show the beginning and the end of the simulation where the drones start from different locations and converge to the same x-coordinate, reaching a vertical arrangement over the final target position on the ground. We may note that the consensus algorithm only controls the movement on the horizontal plane, independently of movements in the vertical directions.

The co-simulation environment is flexible and other co-ordination protocols can be easily analyzed. As an example we have applied the framework to a slight variant of the consensus algorithm above, obtaining an algorithm that performs the task of space coverage along a line segment. We introduced the assumption that the leftmost drone and the rightmost drone do not change their position and they are placed at the endpoints of the line segment. In the

Table 1: Parameters of the scenarios

Scenario	Parameters	Values
Rendez-vous	Duration	40 seconds
	ϵ	$\frac{1}{4}$
	Initial position	{0,1,2,5,10}
Space coverage 1	Duration	20 seconds
	ϵ	$\frac{1}{4}$
	Initial position	{0,1,2,3,10}
Space coverage 2	Duration	20 seconds
	ϵ	$\frac{3}{4}$
	Initial position	{0,1,2,3,10}

following formula, N is the number of drones, min and max are the endpoints of the line segment, $x_d^k(i)$ is the desired position of drone i at step k, and $\epsilon > 0$ is the same parameter of the original algorithm:

$$\begin{cases} x_d^k(1) & = \min, \forall\ k \\ x_d^k(N) & = \max, \forall\ k \\ x_d^{k+1}(i) & = x_d^k(i) + \epsilon(x_d^k(i-1) - x_d^k(i)) + \epsilon(x_d^k(i+1) - x_d^k(i)), \quad i \in [2, N-1] \end{cases}$$

In the following, we show the results of the co-simulation in two scenarios whose parameters are shown in Table 1 (Space coverage). Figures 3a and 3b show the beginning and termination of the co-simulation for the first scenario where the three middle drones started close to each other and end up equally spaced on the x-axis.

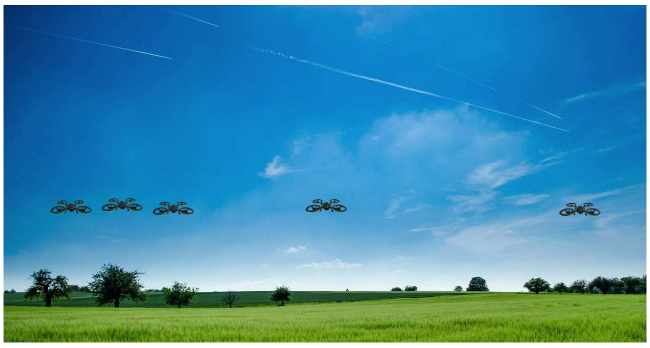
Figures 4 show the termination of the co-simulation for the second scenario where two drones collided and fell to the ground.

The two scenarios show how the value of the parameter ϵ affects the behavior of the drones, which otherwise have the same initial position and are controlled by the same algorithm in the two scenarios. From the simulation, we can see that a large value of ϵ causes the drone coordination to fail. Conditions on the admissible values of ϵ can be determined with the PVS theorem prover, which is the object of further work.

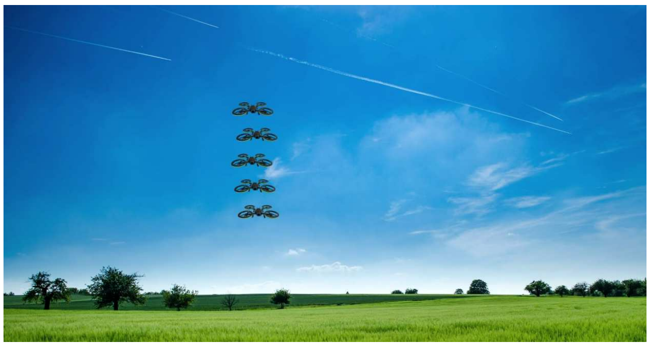
Acknowledgments. The authors would like to thank Paolo Masci for the stimulating discussion on visual interfacing for co-simulation.

References

[1] Jose Joaquin Acevedo, Begoa C. Arrue, Ivan Maza, and Anibal Ollero. Distributed approach for coverage and patrolling missions with a team of het-



(a) Initial position of drones



(b) Final position of drones

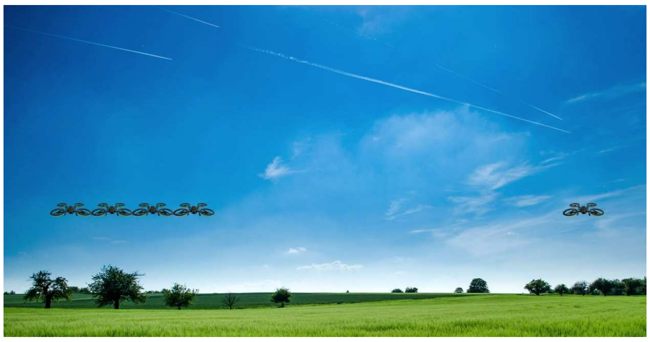
Figure 2: Rendez-vous

erogeneous aerial robots under communication constraints. *International Journal of Advanced Robotic Systems*, 10(1):28, 2013.

[2] Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 173–184. Linköping University Electronic Press, 2012.

[3] Peter Fritzson. Modelica - a cyber-physical modeling language and the openmodelica environment. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1648–1653. IEEE, 2011.

[4] Y. Kuriki and T. Namerikawa. Consensus-based cooperative formation control with collision avoidance for a multi-uav system. In *2014 American Control Conference*, pages 2077–2082, June 2014.


(a) Initial position of drones.

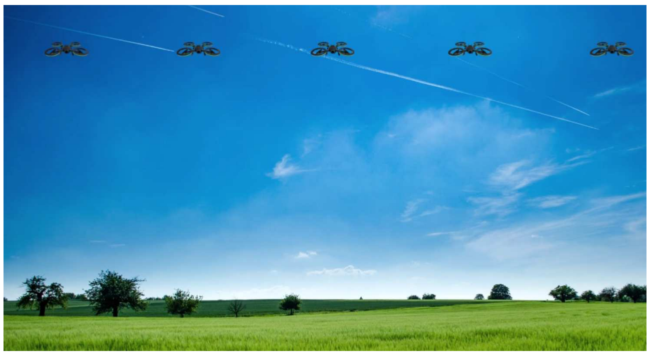

(b) Final position of drones.

Figure 3: Space coverage 1

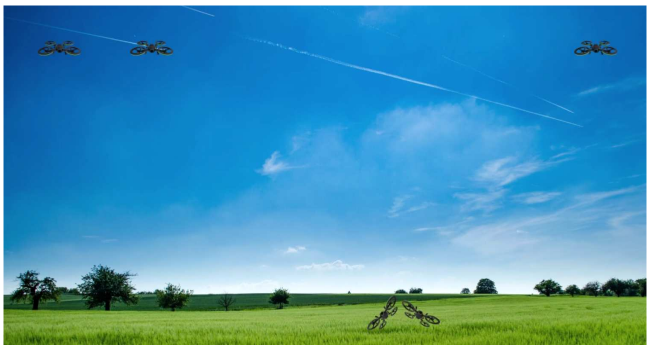


Figure 4: Space coverage 2: final position of drones

[5] Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, Peter Fritzson, Jörg Brauer, Christian Kleijn, Thierry Lecomte, Markus Pfeil, Ole Green, Stylianos Basagiannis, et al. Integrated tool chain for model-based design of cyber-physical systems: The into-cps project. In *Modelling, Analysis, and Control of Complex CPS (CPS Data), 2016 2nd International Workshop on*,

pages 1–6. IEEE, 2016.

[6] Patrick Oladimeji, Paolo Masci, Paul Curzon, and Harold Thimbleby. PVSio-web: a tool for rapid prototyping device user interfaces in PVS. In *FMIS2013, 5th International Workshop on Formal Methods for Interactive Systems, London, UK, June 24, 2013*, 2013.

[7] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan 2007.

[8] S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. PVS Language Reference, Version 2.4. Technical report, SRI International Computer Science Laboratory, 333 Ravenswood Avenue, Menlo Park CA 94025, USA, 2001.

[9] Maurizio Palmieri, Cinzia Bernardeschi, and Paolo Masci. Co-simulation of semi-autonomous systems: The line follower robot case study. In Antonio Cerone and Marco Roveri, editors, *Software Engineering and Formal Methods*, pages 423–437, Cham, 2018. Springer International Publishing.