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

Simulation and Platform Tools to develop safe flock of UAVs : a CPS Application-Driven Research

Laurent Ciarletta[1], Adrien Guenard[2], Yannick Presse[3], Virgine Galtier[4], Ye-Qiong Song[1]
Jean-Christophe Ponsart, Samir Aberkane and Didier Theilliol [5]

Abstract— **Autonomous formation flying (flocking) of UAVs requires coordination of efforts and solutions in multiple research fields (robotics, wireless sensor networks to name a few). The Airborne Embedded auTonomOUs Robust Network of Objects and Sensors (AETOURNOS) project is designed as a federative, fueling platform and global evaluation tool. It has spawned another collaborative project bridging the gap between Computer Science and Automation : 6PO (Cyber-Physical Systems and Cooperative Control of a Fleet of UAVs) . We want to investigate new flocking strategies, mobility-aware communication protocols (and reciprocally: movements influenced by network QoS), and to ease the cooperation between experts, we need co-simulation solutions, and "universal" control interfaces. This article presents our first contribution to this grand scheme: a stand-alone simulation of an autonomous flocking strategy taking into account both the dynamics of the UAVs and their communications, which has allowed us to examine the relation between network unreliability and control computation. To better study the influence of the communication network, we present an architecture coupling a dedicated network simulator with the UAVs simulator, as well as with a real drone piloting interface. The lessons learned help us build a research and action plan for the years to come.**

I. Introduction and Outline

AETOURNOS (Airborne Embedded auTonomOUs Robust Network of Objects and Sensors) is a platform for cross-domain researches around the formation flying of Unmanned Aerial Vehicles (UAVs). It is a multi-team joint-project, mainly within the Loria laboratory but also with other academic institutions such as Supélec or MinesNancy (JaSMinDrone project). It was designed as a catalyst and demonstrator for scientific projects conducted in many different computer science fields. It has been initially composed of teams with interest in: real-time software and advanced networks, multi-modeling and co-simulation, sensors/actuators networks management and security, formal methods applied to safe-by-design software and systems development... While some research works will quickly be giving results, others such as formal methods, are considered in the long run. It has allowed the development of another collobrative research project , 6PO (Cyber-Physical Systems and Cooperative Control of a Fleet of UAVs [1], which combines Computer Science and automation.

These Cyber Physical System (CPS) platforms will offer a full range of testing capabilities from pure simulation to fully-autonomous real indoor and outdoor experiments, including remotely controlled scenarii and co-simulation (i.e. mixing partially implemented code, functional hardware and emulated features and behaviors).

We start this paper with an overview of the context of our project, introducing some sub-goals: flocking (autonomous formation flying)[2], communications QoS (Quality of Service), co-simulation and platform services. We later present our first contributions to this grand scheme: the stand-alone MATLAB/Simulink simulation of an autonomous spring-damper flocking strategy using an advanced model taking into account both the dynamics of the UAVs and their communications. This has allowed us to examine the relation between network unreliability and control computation. To better study the influence of the communication network, we believe that the use of a dedicated network simulator is required, and we present an architecture coupling one with the Simulink-based simulator, as well as with a real drone (AR.Drone compatible) piloting interface. Our results encourage us to push our efforts further, we will conclude by exposing some of our future actions.

II. Context: Research About and Using UAVs

A. A Platform for UAVs in Interaction

The initial idea of AETOURNOS is stemming from the initial Mines-i-Drone (now JaSMinDrone) project to develop a generic experimental and educational platform around UGVs and UAVs. 6PO brings in the equation the necessary control and safety elements with a strong automation background. We are building a set of platforms which can be at the same time demonstrators of scientific production and evaluation environment for research works of various teams of the Loria, CRAN and Supelec laboratories. The scientific questions raised here focus on the CPS and their applications. Those systems consist of numerous autonomous elements in tight interaction with the physical world. Their functioning

*The 6PO part of this work was supported by Region Lorraine

[1]Laurent Ciarletta and Ye-Qiong Song are with Madynes INRIA, Loria, UMR 7503, ENSMN/ENSEM, Lorraine University, Nancy, France firstname.name@loria.fr

[2]Adrien Guénard is with Loria, Nancy, CPE, Lyon, France firstname.name@alerion.fr

[3]Yannick Presse is with Loria, Madynes INRIA, Nancy, France firstname.name@inria.fr

[4]Virginie Galtier is with Supélec UMI 2958, Metz, France firstname.name@supelec.fr

[5]Jean-Christophe Ponsart, Samir Aberkane and Didier Theilliol are with Centre de Recherche en Automatique de Nancy, Universit de Lorraine, CNRS UMR 7039, F-54506 Vandoeuvre-les-Nancy, France firstname.name@univ-lorraine.fr

[1]Systèmes Cyber-Physiques et Commande Coopérative Sûr de Fonctionnement pour une Flotte de Véhicules sans Pilote

[2]"étourneau" is French for "Common Starling", a bird specie known for forming large flocks.

requires a strong coupling between software implementations and technical devices (harware, sensors/actuators/controlers) in order to achieve safe execution. The collective movements of a flock of flying communicating robots / UAVs, evolving in potentially perturbed environment constitute a good example of such a system. Indeed, if we look at the level of each of the elements playing a role into this system, a certain number of challenges and scientific questions can be studied: respect of real-time constraints of calculations for every autonomous UAV and for the communication between the robots, design of individual, embedded, distributed or global management systems, development of self-adaptative mechanisms, design of algorithms for collective movement etc... Furthermore, each of these questions can not be considered in isolation, but rather must takes into account the whole ecosystem: in the end, solutions have to contribute to the global operation of the system! The next sub-sections detail four directions towards which AETOURNOS aims to propose advances and where 6PO contributes.

B. The Flocking Behavior

The collective movements of robotic agents have already been studied according to various approaches. Among those are the bio-inspired approaches of natural systems. These works focussed on the collective movement of the human crowds, on the thick clouds of birds, on fish shoaling or on the movements of bacterias. Introduced by Graig Reynolds [14] for problems of simulation and display, these works interested as well biologists, physicists, IT specialists or still roboticists. Today, the physicists are especially active in this domain and we can mention in particular the recent works of Vicsek [18]. Moreover, we base our collective movements experience on several studies of robots platooning, like the CRISTAL project [8]. In this work, we mainly focus on birds movements, called flocking. A flock is a group of moving agents clustered together. In his works, Graig Reynolds suggested that simple rules used on a flock of mobile agents lead to good results in term of collision avoidance, velocity matching and flock centering. The three main rules are the alignment to make the agents moving in the same direction, the separation to avoid collisions and the cohesion to maintain the agents with an acceptable distance in between. These rules can be seen as forces applied on the mobile agents. Figure 1(c) shows these attractive and repulsive forces on a moving agent.

In our project, we plan to use a combination of inertial measurement, positioning information and networking status to implement the flocking behavior.

C. Networking and Communications

Our flock of UAVs uses MANET (Mobile Ad-hoc Network) communications to exchange position (and other) information between each other that will be used as inputs to individually compute their movements. Eventually a coherent collective movement should emerge. For that coordination to be reliable, a given level of communication QoS is required. Many wireless sensor network protocols are QoS-aware [11]. But because of their movements, UAVs might end up partitioned in multiple groups, in which case auto-repair mechanisms must be implemented so that communications are maintained within each of them. And once the separated groups join again, global connectivity must be re-established with the whole fleet. Existing works need to be extended to support those additional constraints: first, current proposals, such as those from the IETF 6lowPan group, only partially integrate mechanisms of self-healing, essentially because they consider a restrictive communication model, namely from multi-sources to a unique sink, that would work for a system with a ”leader”, while we will eventually need multipoint-to-multipoint communications support in the fully distributed scenario. Second, they consider situations where nodes failure results in a network partition restricted to 2 groups [2], while we need to consider situations with multiple nodes failures leading to a larger number of partitions. Lastly, existing work usually propose solutions at the routing level whereas mobility could be the solution instead of being the issue: by moving adequately one node, it can be positioned as to fill the communication gap between partitions. Two kinds of mobility must be examined: sequential node by node movements [1] and global node movements [4]. To design new solutions, routing and mobility must be considered jointly and realistically, and one objective of AETOURNOS is to contribute to those aspects. It will help us study how the unreliability of mobile nodes wireless networks gives real challenges for control application development.

D. Simulation/Co-simulation

In the case we study here, CPSs are deployed over WSANs, combining both sensors and robots interconnected by wireless networks. This kind of systems have to meet specific requirements and have to provide specific QoS. To ensure the system meets the required properties, we model it for a priori evaluation. The difficulty in simulating these systems is mainly due to the need of modeling both the continuous part (robots kinematics, mobility functions...) and the discrete part of the system (control algorithms, communication protocols...). Regarding the particular case of agent flocking, numerous simulations have been developed under sharp hypothesis: usually the agents are considered as point masses, no communication occurs between them (or it is immediate) and their positions are well known. In our simulation, we want to implement a more realistic model of the entire system. As a result, we simulate the robotic agents with a dynamic model, exchanging information with each other through a network. One advantage of an accurate simulation is that it offers possibilities for co-simulation or hardware-in-the-loop simulation. Yet developing an inclusive, stand-alone model is tedious since each aspect of the system must be modeled from scratch; yet models usually already exist in each community, and are polished by various experts. It would be more efficient to use those models rather than re-develop them. For instance, the network part of our simulation was initially quite limited, so we have combined

it with a dedicated network simulator (OMNeT++). The difficulty is that models, and associated tools, are not designed to cooperate. AETOURNOS will assess some of the standards for model-exchange and co-simulation (such as FMI, the Functional Mockup Interface [5] or HLA, the High Level Architecture [16]) and will extend some of our previous work done on building complex simulation as a society of simulators and on novel mobility models for network evaluation using MABS (Multi-Agent Based Simulation) [10]. We will apply our framework AA4MM as a modeling and orchestration paradigm [17].

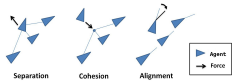
E. Platform and Services

In parallel with the formation flying goal, the AETOURNOS project is designed as an application driven research platform. We have a dedicated space (length 8m; width 4m; height 5m), the "cage", protected by a classical "UAV net" (see figure 1(a)). Since we want autonomous and collaborative flight behaviors, we won't base our platform on external solutions such as "motion capture" to localize and control our quadrotors. As far as localization is concerned, we are developing a combination of wireless sensor techniques, with both in house solutions and commercial based (Fireflies RTLS ©), and vision-based (cameras, MS Kinect©) that will allow indoor or outdoor, relative or absolute positioning.

Multirotor (quadrotor) platforms have been chosen for their good maneuverability and payload capacity. We initially used two kinds of robots, the Parrot AR.Drone , developed for entertainment and the Asctec Pelican , developed for research platforms. We have invested in hexarotors (Astec Fireflies. They have a failsafe mechanism that can handle single engine failure. One long term goal is to provide autonomous outdoor flying of our "squadron" (figure 1(b)).

(a) Flying in the "cage"

(b) The "squadron"



(c) Classical model of flocking forces

Fig. 1: Real quadrotors (and hexarotor), cage and squadron

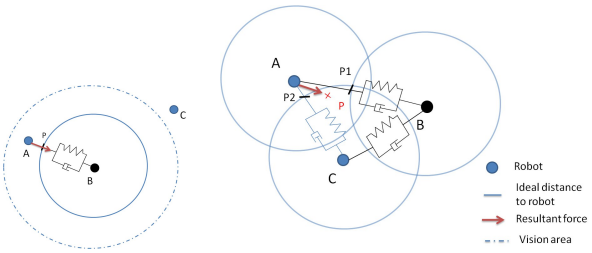
Most robots and UAVs usually come with dedicated control interfaces (iOS and Android applications for AR.Drone for instance) and SDKs which allow tailored developments. Unfortunately, these solutions are often product-specific, which makes maintaining a platform composed of several brands of robots more difficult. Community tools, such as ROS (Robot Operating System, [15]), provide a unified way to control these robots and offer various programming language interfaces. But it is at the cost of complexity (installation, configuration and usage). Since many people are familiar with Java, including students working on short-term projects on our platform as well as researchers from various computer science fields, we have chosen it in 2010 to initiate our development. That's one of the reasons why a Java interface for controlling a drone was first developed by students from ENSMN (MinesNancy) as a pedagogical assignment [3]. In addition to providing a Java library for the remote control of multiple AR.Drones using the SDK from Parrot, the students have developped a set of demos (attitude control of one multirotor with a second one, robot following another robot thanks to the "tag" detection function of the AR.Drone firmware). The student video can be found at: `http://www.youtube.com/watch?v=4EQum6BXZls`.

III. Simulation (Simple UAVs and Networking Interweaved Behaviors)

A. The Spring-Damper Flocking Model

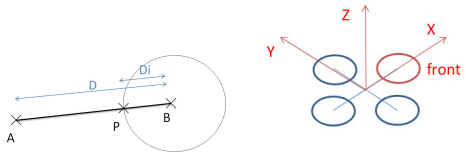
The idea is to implement a first model of flocking behavior with several robots. The implementation here is based on a simplified model of Graig Reynolds' one. The objective is to create a collective movement of robots, globally steered by a leader one. The two main purposes can be summarized into two goals: each robot has to follow the leader within the flock and avoid collisions with other robots. First of all, the robots we use here are quadrotors. That means that we use holonomic robots, able to move in any direction with any orientation. The first simplification of the model is to consider to ignore the alignment rule. Indeed, if a robot follows the leader, the directions would be close. The second simplification is made on the two forces of cohesion and separation. We consider here that these two forces are opposite and collinear. That means that we can regroup the forces into a single resultant one. The idea is to consider a mechanic spring-damper model giving a visual representation of the interaction between two robots. For each robot we define two distances: the vision distance and the ideal distance. The first one defines the area within which one robot interacts with others. Then, a single robot will interact only with the directly surrounding robots. The ideal distance Di represents the target distance for robots surrounding another one avoiding collision and maintaining an acceptable flock. In our model, this distance corresponds to the equilibrium length of the spring-damper system. If two robots are too close to each other, the spring compression generates a force to separate them proportionally to the compressed length. Also, if two robots are too far from each other, the system will tend to reduce the distance in between. The figure 2(a) represents this interaction between two robots.

To compute this system, since the force generated by the spring is proportional to the error distance regarding to the equilibrium length, this system can be modeled by a proportional corrector. Similarly, the effect of the damper is proportional to the velocity of the moving robot. The



(a) Two robots interaction

(b) Multiple robots interaction



(c) Determination of the closest point. $P = B - \frac{\vec{AB}}{D} * Di$

(d) X, Y and Z axis

$$\vec{AB} = \vec{AP} + \vec{PB} \quad \text{(e)}$$

$$B - A = P - A + \frac{\vec{AB}}{\|\vec{AB}\|} * \|\vec{PB}\| \quad \text{(f)}$$

Fig. 2: Spring-damper flocking model, Determination of the closest point (bottom)

spring-damper system is then equivalent to a proportional-derivative (PD) controller on robots positions. For two robots, the setpoint for this controller is the closest point from one robot which is at the Di distance from the other robot (Point P on figure 2(a)). To compute the controller, we calculate the distance between one robot to this point P. Considering two robots A and B separated by a distance D (figure 2(c)), we can find the coordinates of the point P. From this, we can compute a position PD controller to have a control loop feedback which tends to maintain robots at the desirable distance Di. Moreover, to eliminate steady-state errors and compensate perturbations, we introduce a third term, proportional to the integral of the position error. Finally, to maintain robots at a desired distance, we use a classic proportional-integrate-derivative (PID) controller. Considering a robot and a desired position we define by e the position error. We can compute the acceleration of its robot using this control loop. Let's consider x the position of the robot and Kp, Kd, Ki the proportional, derivative and integral gains of the PID controller. We can use this control law to compute the robot acceleration:

$\ddot{x} = Kp * e + Kd * \dot{e} + Ki * \int e$

For the usual case of a robot interacting with more than one other, this controller provides an easy way to compute a new control law. For one single robot, we calculate the desired position regarding the position of each robot surrounding it. Then, the desired position we use for the setpoint of the position controller is the barycenter of all the positions calculated for each couple of robots. This is shown in the figure 2(b). In this example, to compute the setpoint position of robot A: we calculate first the desired position considering only interaction with robot B. This gives the point $P1$. The same principle for robot C gives the point $P2$. We compute then the barycenter of $P1$ and $P2$, which gives the point P, the setpoint position for the PID controller of robot A. The same is done for robots B and C. The robots will organize themselves into a minimum energy configuration, regarding the energy stored in the mechanic spring-damper system. Another advantage of this simple algorithm, is that it is possible to easily define a leader robot for the flock. Indeed, it is possible to give to each robot a weight value taken into consideration for the setpoint positions barycenter calculation. A leader robot is defined with a great weight value.

B. The UAV Model and first tests

To perform the simulation task, we use our own framework, Samovar [9]. It is based on MATLAB/Simulink augmented by a real-time systems simulator, the TrueTime [7] toolbox. This allows the modelling of both physical systems, execution platforms and network protocols. It provides a three dimensional visualization with the Irrlicht engine embedded in Simulink, and offers the possibility for robots to interact with virtual reality environments through sensors. The advantage of this framework is to be able to study different aspects of such systems. Quadrotor robots are implemented with a realistic dynamic model, controllers are modeled taking into account real-time constraints like execution time or scheduling policy and networks deal with MAC and routing protocols or unreliability like time delays or message losses. That allows accurate robotics, control and communication studies of the system.

A realistic model of quadrotor is then used to simulate one robot behavior. As in [6], dynamic and kinematic models have been implemented, taking into account mechanics and aerodynamics constraints. We model the quadrotor with four rotors in a cross configuration. One rotor is considered as the front of the robot, and three axis are then defined as shown in figure 2(d): the X axis from back to front, Z axis vertically and Y along the rotor axis perpendicular to X axis. Four variables can be tuned to control the robot. The throttle U, leads to a vertical force to set the height of the quadrotor. The throttle is controlled by an equal variation of the four propeller speeds. The three other commands roll (Φ), pitch (Θ) and yaw(Ψ) are respectively related to rotation angles about the X, Y, and Z axis. We consider Ω_1, Ω_2, Ω_3 and Ω_4 respectively the front, right, rear and left propeller speeds, and Ixx, Iyy and Izz the moments of inertia around the X,Y and Z axis. A PID controller uses the acceleration on Z axis and the rotation accelerations to compute the four control commands.

$$\begin{cases} \ddot{\Phi} = \dfrac{lb*(-\Omega_2^2+\Omega_4^2)}{Ixx} \\ \ddot{\Theta} = \dfrac{lb*(-\Omega_1^2+\Omega_3^2)}{Iyy} \\ \ddot{\Psi} = \dfrac{d*(-\Omega_1^2+\Omega_2^2-\Omega_3^2+\Omega_4^2)}{Izz} \\ \ddot{Z} = -g+(cos(\Theta)cos(\Phi))\dfrac{U}{m} \\ U = b*(\Omega_1^2+\Omega_2^2+\Omega_3^2+\Omega_4^2) \end{cases} \quad (1)$$

[6] gives these accelerations equations (equations 1) with b and d aerodynamic thrust and drag factors, and l the distance between the center of the quadrotor and the center of a propeller.

$$\begin{cases} \ddot{X} = (sin(\Psi)sin(\Phi)+cos(\Psi)sin(\Theta)cos(\Phi))\dfrac{U}{m} \\ \ddot{Y} = (-cos(\Psi)sin(\Phi)+sin(\Psi)sin(\Theta)cos(\Phi))\dfrac{U}{m} \\ \ddot{Z} = -g+(cos(\Theta)cos(\Phi))\dfrac{U}{m} \end{cases} \quad (2)$$

To compute the robot velocities with regard to a fixed frame, we use equations (equations 2), with m the mass of the robot.

Dynamics are also taken into account, by modelling motors with first-order functions. Noise and delays are introduced in sensors feedback loop, to design a more realistic model [22].

A specific UAV control law has been developed that is based on a feedback linearization approach. This approach involves coming up with a transformation of the nonlinear system into an equivalent linear system through a change of variables and a suitable control input. Feedback linearization may be applied to nonlinear systems of the form.

$$\dot{x} = f(x)+\sum_{i=1}^{m} g_i(x)u_i \quad (3)$$

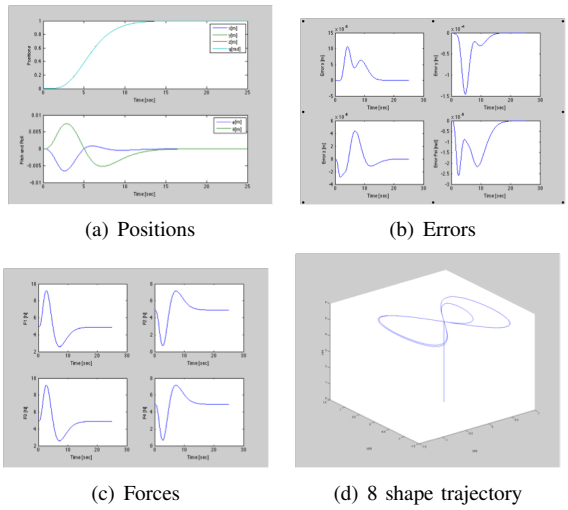
where $x \in Rn$ is the state vector, $u \in Rm$ is the vector of inputs, and $y \in Rp$ is the vector of outputs. The goal is to develop a control input $u = a(x)+b(x)v$ that renders a linear input-output map between the new input v and the output. An outer-loop control strategy for the resulting linear control system can then be applied ([xx]).

Once the system is linearized, we get:

$$\begin{cases} x^4 = v_1 \\ y^4 = v_2 \\ z^4 = v_3 \\ \phi^2 = v_4 \end{cases} \quad (4)$$

thus we have 3 fourth order integrators, and one second order integrator. In order to stabilize the outputs, a pole placement method will be employed. control technique.

From a physical point of view it means that when one of the rotor fails, the inner control law stabilizes roll, pitch and altitude, while the outer control law exploits the near hover condition to slowly change the pitch and roll angles in order to move the vehicle to a desired position in space.



(a) Positions (b) Errors

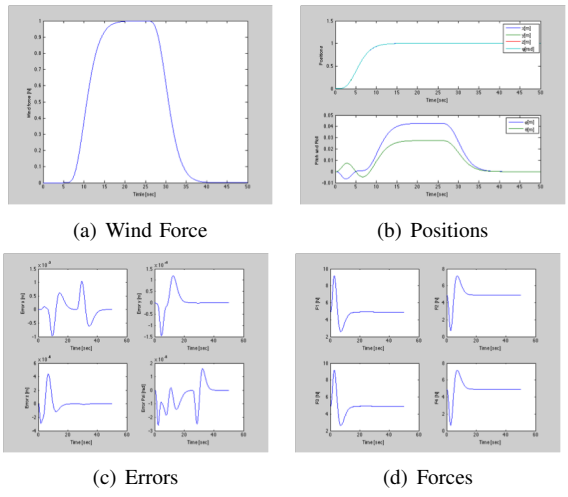
(c) Forces (d) 8 shape trajectory

Fig. 3: Feedback linearization results and 8 shape

Another test has been realized by considering the 8 shape trajectory. The quadrotor presented a steady trajectory (Figure 3(d)).

To test the robustness of the Feedback Linearization technique, the previous simulation was repeated but this time a wind force was introduced as a disturbance between t=5s and t=40s. The wind force is defined as represented in the following figure ((Figure 4(a)).

We can see that during the 20 to 25 seconds time interval, while wind is still blowing through the quadrotor (Figure zz), x, y, z, ϕ, θ, ψ, are stabilized (Figure 4(b)), the errors are set to zero (Figure 4(c)) and the forces generated by the actuators are acceptable (Figure 4(d)). This indicates a good robustness of the control technique [21].



(a) Wind Force (b) Positions

(c) Errors (d) Forces

Fig. 4: Feedback linearization results, a wind disturbance

C. Communication Between Robots

To compute the control law for one robot, it is necessary to know its own position and the one of its surrounding robots.

In a first approximation, we consider that one robot is aware of its position. The positions of the other robots are transmitted through a wireless network. This introduces challenges of communication between mobile nodes, in particular for routing. We use the AODV (Ad-hoc On-demand Distance Vector) dynamic routing protocol (given by the TrueTime library) to transmit information.

D. Experiments

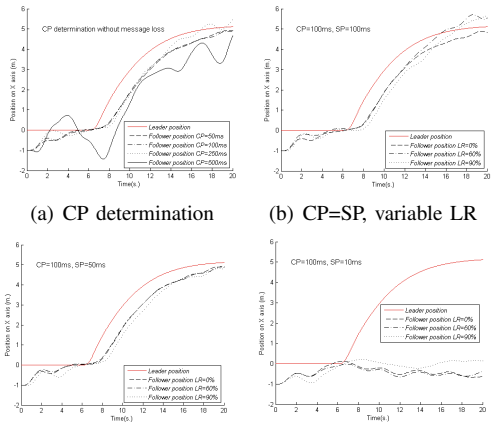
We have implemented the flocking algorithm in the Samovar framework applied to a flock of seven robots (a leader is selected and manually controlled). The goal is to make the flock of autonomous robots adapting to the leader position and velocity, in order to follow it, avoiding collisions. Before implementing the flocking algorithm on a concrete platform, it is necessary to be sure about feasibility and robustness of the algorithms. The spring-damper flocking algorithm is implemented on all the robots controller. They are also exchanging positions with wireless messages. To compute the PID setpoint positions of one robot, the weight value attached to the leader is equal to the sum of the weight values attached to the other ones in its vision area.

Our scenario: the leader takes off, waits for 3 sec., giving time for the flock self-organization around it (figure 6(a)). Then the leader moves along the X-axis during 10 sec., finally stabilizing. We see that when the leader is static, the circular flock shape around it corresponds to the minimum energy configuration (in conformance with the spring-damper model equivalence). When the leader moves, the other robots follow and avoid collisions (figure 6(b)).

Two parameters have to be tuned: the Control Period (CP) and the Sending Period (SP). CP is the time interval between 2 commands sent from the controller to the engines. SP is the inter-message period. To determine a suitable CP, our first experiment used a lossless communication model. As seen on figure 5(a), a CP of 50 or 100ms gives an optimal following behavior. While a smaller period doesn't improve it, it implies more computations. A larger period exhibits oscillations. Therefore we picked a CP of 100ms for the remaining experiments.

In CPS, we can expect noisy environments, leading to message losses for example. In order to keep an efficient control, we need fresh enough position data. With the same scenario, we have introduced some packet loss (variable Loss Rate : 0, 60 and 90%). Sending data at the needed rate for the controller computation (CP=SP), the flocking behavior degrades (see figure 5(b)). With a doubling of the sending rate, which seems appropriate, even a 90% LR gives good results, because of the redundancy of the information and the stability of the controller (see figure 5(c)). But we see that with 10 times more messages, we are actually overloading the network, and our system exhibits an unacceptable behavior (see figure 5(d)).

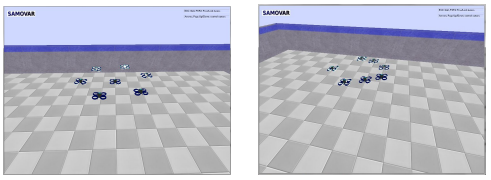
This simple algorithm provides good results with some remaining problems. Indeed, the performances are acceptable for normal functionning mode: robots are safely flocking



(a) CP determination (b) CP=SP, variable LR

(c) CP = 2 SP, variable LR (d) CP = 10 SP, variable LR

Fig. 5: Quadrotor flocking simulation, CP: Control period, SP: Sending period, LR: Loss Rate



(a) Leader static: stabilized flight (b) Leader moving

Fig. 6: Flocking simulation, Squadron following leader

around the leader. Yet, in some configurations, some collisions can occur. For example, very fast changes in the leader trajectory may lead to unsafe behaviors.

IV. Multi-simulation (Coupling Simulators) and Co-simulation (Mixing Real Hardware and Software and Simulation): First Achievement

The overall goal here is to be able to model the communication part, the control part, physical world, and integrate real pieces of software and hardware.

A. JaSMinDrone: Piloting Real Drones

The API mentionned in section II-E has been updated and is now called JaSMinDrone (Java Supélec Mines Drone interface) [13]. As shown in figure 7 and in "Real world" upper box from figure 8, this interface allows the user equipped with a computer with a WiFi interface to send commands to the drone as well as to display information received back from the drone: video and navigation data (attitude, battery level, detected tag position, etc.).

This interface provides an easy way to achieve scenarii involving several drones (one computer can control multiple drones). In addition this software provides an improved GUI for interactive piloting, and a separated library to script piloting commands (which makes it easier to repeat experiments, and to execute a strict flight plan). It has recently been updated to support AR.Drone V2.

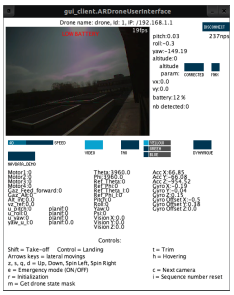


Fig. 7: JaSMinDrone graphical user interface

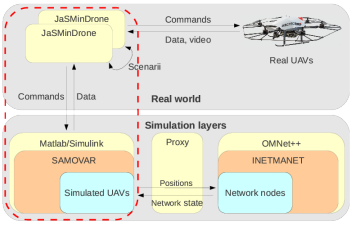


Fig. 8: The pilot interface, UAV and communication network simulation tools coupling architecture

B. Connecting JaSMinDrone to SAMOVAR: Piloting Simulated Drones

As shown by the red-doted part of figure 8, we have connected JaSMinDrone with Samovar. This allows to control simulated drones with the same interface as real ones: a graphical interface (JaSMinDrone offers a user-friendly display of the drones navigation data whereas Samovar requires the addition of scopes to visualize them), or a scripting interface (in Samovar the leader drone can be controlled only interactively, using the keyboard while the JaSMinDrone scripting function allows the repetition of experiments).

Communication between both modules uses a network socket. The only extension to JaSMinDrone is the ability to specify the socket from which sending commands and receiving data: connection to the real drone or to Samovar. The drone model in Samovar is kept unchanged, modifications are limited to receiving the commands from the socket instead of the keyboard and adequately editing navigation data to match the expected, real drone like format.

The user experience is similar to controlling a real drone, except that some navigation data are not available (motors pwm, gyroscopic data, etc.) and neither is the video.

C. Connecting SAMOVAR with OMNeT++: Adding Advanced Networking Models

A shortcut of the scenario described in section III-D is the use of the TrueTime toolbox. Even though its integration was immediate, it is not the favorite framework for network protocol designers; beside, not many protocols are available (AODV and TCP are only provided as examples). In order to study the influence of different network protocols on the overall operations of the drones, the use of a dedicated and widely-accepted network simulator such as OMNeT++ or ns-3 is an asset. Consequently, we have connected Samovar to OMNeT++ as depicted on the "Simulation layers" lower box in figure 8.

Cooperation between these modules made it possible to test a new scenario where drones movements are influenced by communication capabilities among the fleet: a drone takes off and flies in the overall direction of another one, testing its network reachability on a regular basis using ICMP echo requests. As soon as the drone reaches the vicinity of the other drone, the latter issues a simulated ICMP echo reply message that triggers a change in the flight direction of the first drone (see figure 9).

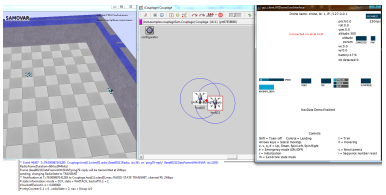


Fig. 9: Coupling of UAVs and network simulators

For pragmatical reasons, we haven't use a dedicated co-simulation solution such as HLA or MSI (Multi-Simulation Interface). Yet to preserve a loosely coupling, we have introduced a proxy connected to the simulators by sockets; it will make it easier in the future to transparently replace one simulator by another (ns-3 instead of OMNeT++, Gazebo instead of Samovar for instance). The previous Samovar model has been augmented along two lines: (1) a control model implements the repulsion behavior described previously, and (2) a new socket is used to send drones positions to the proxy and to receive network reachability information from the proxy. On the OMNeT++ side, we have used the INETMANET framework which includes models of MANET nodes equipped with a complete TCP/IP stack. A new mobility model had to be developed to take into account the positions information received from the proxy, and to send the information about network reachability to the proxy.

D. Putting It All Together and Outlook

By composing the former pieces we were able to build a demonstrator in which a real drone controlled by JaSMinDrone interacts with a simulated drone and influences its flying behavior based on simulated wireless network reachability information.

V. What's Next?

a) Larger testbed and more integrated tools: For the 2014-2015 period, we will work on building a new test environment (a larger cage) and on developing a set of tools (SDK, tutorials, dedicated virtual appliances). This will make the platform usable by a larger community, which is one of our main goals. This is funded by the INRIA ADT

program under the R2D2 (Resources and Research for Drone Development) name.

b) Improved flocking algorithm and custom communication stack: Our flocking algorithm can be improved by numerous means. To avoid the risk of collisions, a solution would be to make the PID controllers respond faster but it will generate unacceptable oscillations. Another solution is to communicate information like motor commands or attitude between the robots. Using this can lead to anticipate robots behavior and make the system more reactive. This introduces wireless communication challenges. The alignment force can also be implemented using attitude and velocity information communicated by robots. We will study and compare several ad hoc and sensor protocols (layer 2 and 3) for communications between quadrotors. We plan to design cross-layer protocols for information dissemination and control.

c) Integration of formal specification: The work already done on ground vehicle platooning [12] could be extended to 3D flocking (from 2D properties such as having a minimal distance between elements of the platoon). One of the central point of interest is to specify and verify properties about "What characterizes a flock?" . A simulator code generator from formal specification will be plugged into our platform through the API.

d) Absolute vs relative, indoor vs outdoor localization: Some hypothesis can not be made anymore for an implementation on a concrete platform. One of the most important challenge to be solved concerns the localization of the robots. Our first implementation considers that robots are aware of their positions. The challenge is to make an autonomous flock of robots. That means that no global localization solution can be used, except GPS for outdoor applications which can not be accurate enough for an acceptable flock. In fact, at first, only distances between robots are needed to compute this flocking algorithm, although dynamic or absolute orientation given by a gyro or compass would help in improving. This is a useful simplification of this problem which can be solved by the use of embedded relative localization systems like cameras, lasers or kinect-like systems.

VI. Conclusion

We have presented the AETOURNOS and its 6PO sibling projects and some initial results. They are scientific platforms for both conducting and demonstrating research works. We have described this as a multi-team catalyst for cross-domain researches around the common topic of Cyber Physical Systems. Our first challenge application is UAVs flocking. We began our study with a pure simulation, taking into account dynamics of multirotors and communication protocols. We presented some results on a predefined scenario to find suitable values for the message sending period and the control period, according to the message loss rate over the network. Then we added the first co-simulation results that we get, coupling real and simulated entities, and coupling different simulators. This led to the definition of our next steps, highlighting numerous challenges to tackle for implementation as one more usable and integrated platform.

References

[1] Abbasi A, Younis M, Baroudi U (2010) Restoring connectivity in wireless sensor-actor networks with minimal topology changes. In: Communications (ICC), 2010 IEEE International Conference, pp 15

[2] Akkaya K, Senel F, Thimmapuram A, Uludag S (2010) Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. Computers, IEEE Transactions 59(2):258271

[3] Andurand P, Hoelter J, Plachot M, Tissot B (2011) Plateforme de développement robotique en Java pour lAR.Drone. Tech. rep., ENSMN

[4] Basu P, Redi J (2004) Movement control algorithms for realization of fault tolerant ad hoc robot networks. Network, IEEE 18(4)

[5] Blochwitz T, Otter M, Arnold M, Bausch C, Clau C, Elmqvist H, Junghanns A, Mauss J, Monteiro M, Neidhold T, Neumerkel D, Olsson H, Peetz JV,Wolf S (2011) The functional mockup interface for tool independent exchange of simulation models. In: modelica2011

[6] Bresciani T (2008) Modelling, identification and control of a quadrotor helicopter. Master Thesis p 213

[7] [7] Cervin A, Ohlin M, Henriksson D (2007) Simulation of networked control systems using TrueTime. In: 3rd International Workshop on Networked Control Systems: Tolerant to Faults, Nancy, France

[8] Havet L, Simonot-Lion F (2009) Timing properties requirements and robustness analysis of a platoon of vehicles. In: 8th IFAC International Conference on Fieldbuses and networks in industrial and embedded systems - FeT2009, Ansan, Korea, Republic Of

[9] [9] Havet L, Guénard A, Simonot-Lion F (2010) Samovar : An evaluation framework for real time applications deployment over WSANs. In: 15th IEEE Int. Conference on Emerging Technologies and Factory Automation - ETFA 2010, Bilbao, Espagne

[10] [10] Leclerc T, Siebert J, Chevrier V, Ciarletta L, Festor O (2011) Multi-modeling and co-simulation-based mobile ubiquitous protocols and services development and assessment. In: 7th Int. ICST Conf. on Mobile and Ubiquitous Systems - Mobiquitous, Sydney, Australia

[11] [11] Li Y, Chen CS, Song YQ, Wang Z (2007) Real-time QoS support in wireless sensor networks: a survey. In: 7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT2007, Toulouse, France

[12] [12] O Simonin AS A Lanoix, Charpillet F (2011) Specifying in B the Influence/ Reaction Model to Study Situated MAS: Application to vehicles platooning. In: V2CS 1st International workshop on Verification and Validation of multi-agent models for complex systems, Paris, France

[13] [13] Presse Y (2012), Etude et conception dune plate-forme de multisimulation et de controle de drones Design of a plateform for drone multisimulation and control. Engineering master thesis, CNAM, CNAM de Lorraine, Metz, France

[14] [14] Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model. In: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, SIGGRAPH 87, pp 2534

[15] [15] ROS (2013) Ros (robot operating system). www.ros.org

[16] [16] S I S Committee et al (2000) IEEE standard for modeling and simulation (m&s) high level architecture (hla). IEEE std 1516-2000, 1516.1-2000, 1516.2-2000

[17] [17] Siebert J, Ciarletta L, Chevrier V (2010) Agents and artefacts for multiple models co-evolution. building complex system simulation as a set of interacting models. In: 9th Int. Conference on Autonomous Agents and Multiagent Systems - AAMAS, Toronto, Canada

[18] [18] Vicsek T, Zafeiris A (2012) Collective motion. Physics Reports 517(34):71 140, DOI http://dx.doi.org/10.1016/j.physrep.2012.03.004 0370157312000968

[19] J. Ghandour, Fault-tolerant flight control techniques: application to a quadrotor UAV testbed, Master Thesis, University of Lorraine, 2013.

[20] A. Chamssedine, Y. Zhang, C-A. Rabbath, C. Join, D. Theilliol Flatness-based Trajectory Planning/Re-planning for a Quadrotor Unmanned Aerial Vehicle IEEE Transactions on Aerospace and Electronic Systems, Vol. 48, No. 4., pp. 2832-2848, 2012

[21] J. Ghandour, Fault-tolerant flight control techniques: application to a quadrotor UAV testbed, Master Thesis, University of Lorraine, 2013.

[22] A. Chamssedine, Y. Zhang, C-A. Rabbath, C. Join, D. Theilliol Flatness-based Trajectory Planning/Re-planning for a Quadrotor Unmanned Aerial Vehicle IEEE Transactions on Aerospace and Electronic Systems, Vol. 48, No. 4., pp. 2832-2848, 2012