3

Field Programmable Gate Array (FPGA) for Bio-inspired visuo-motor control systems applied to Micro-Air Vehicles

Fabrice Aubépart, Julien Serres, Antoine Dilly, Franck Ruffier and Nicolas Franceschini

Institute of Movement Science., University of the Mediterranean and CNRS France

1. Introduction

Nature provides us with many examples of ingenious sensors and systems at the service of animal behavior, which can be transferred into innovative systems for the control of Micro-Air Vehicles (MAVs). Winged insects demonstrate daunting behaviors in spite of the poor resolution of their visual system. For more than 100 million years, they have been navigating in unfamiliar 3D environments by relying on *optic flow* (OF) cues. To sense and react to the flowing image of the environment, insects are equipped with smart sensors called Elementary Motion Detectors (EMDs) that can act as optic flow sensors (figure 1). The principle of our bio-inspired optic flow sensors is based on findings obtained at our laboratory on the common housefly's EMDs, by performing electrophysiological recordings on single neuron while applying optical microstimuli to two single photoreceptors cells within a single ommatidium (Franceschini, 1985, Franceschini et al. 1989).

The OF-field gives the *angular velocity* (in rad/s) at which any contrasting object in the environment is moving past the eye (Koenderink, 1986). One lesson we have learned from insects is that they are able to navigate swiftly through the most unpredictable environments without using any velocimeters or rangefinders. Insects rely on optic flow to avoid collisions (Collett, 1980; Wagner, 1982; Tammero and Dickinson, 2002), to follow a corridor (Kirchner and Srinivasan, 1989; Baird et al. 2005; Ruffier et al., 2007; Serres et al., 2007; Serres et al. 2008), to follow the terrain (William, 1965; Srygley and Oliveira, 2001), to fly against wind (Kennedy 1939, Kennedy 1951), and to cruise and land (Srinivasan et al., 1996), for example.

Interestingly, insects seem to maintain a constant optic flow with respect to their surrounding environment while cruising and landing (Kennedy, 1951; David, 1978, Srinivasan et al. 1996, 2000). Several MAV autopilots were built in recent years which show how insects could achieve this feat by using a feedback control system called the optic flow regulator (Ruffier and Franceschini, 2003, 2005, Serres et al. 2008). Future MAVs' visual guidance systems may have to incorporate optic flow sensors covering various parts of the visual field, like insects do (Figure 2).

The biorobotic approach developed at our laboratory over the past 20 years enabled us to construct several terrestrial and aerial robots based on OF sensing (Pichon et al., 1989, Franceschini et al., 1992, 1997; Mura and Franceschini, 1996; Netter and Franceschini., 2002,

Ruffier et al., 2004). The robot Fly ('le robot-mouche') started off as a small, completely autonomous (terrestrial) robot equipped with 114 optic flow sensors. This robot was able to steer its way to a target through an unknown field of obstacles at a relatively high speed (50cm/s) (Franceschini et al. 1992, 1997). Over the last 10 years, we developed small (mass < 1kg) optic flow based aerial demonstrators with limited degrees of freedom (Viollet and Franceschini, 1999, 2001 ; Netter and Franceschini, 2002; Ruffier and Franceschini, 2003, 2005; Kerhuel et al., 2007; Serres et al., 2008,) called FANIA, OSCAR, OCTAVE, and LORA, respectively.

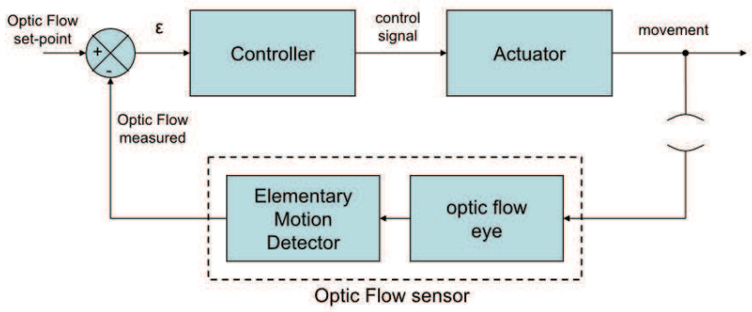


Figure 1. Principle of an optic flow regulator. The feedback control loop aims at maintaining the OF measured constant and equal to an optic flow set point

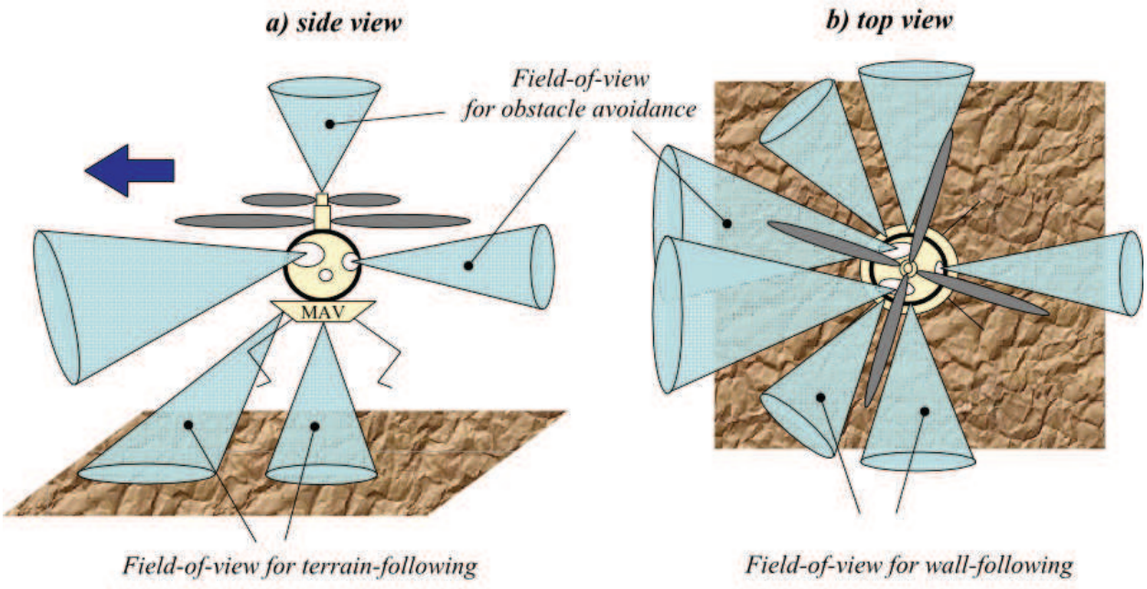


Figure 2. Several optic flow sensors covering various, strategic fields of view for an insect-like visual guidance of Micro-Air Vehicles (MAV)

In addition to visuo-motor control loops, inner control loops are necessary (i) to lock the micro-aircraft in certain desired attitudes, and (ii) to improve the control performances and robustness.

(i) Attitude control systems laid out in parallel with the *visuo-motor control loops* may use inertial and/or magnetic angular sensors (Ruffier et al., 2005; Viollet et al., 2001; Serres et al., 2008)

(ii) Control speed and accuracy can be improved with advanced actuators and more classical control loops based on proprioceptive sensors.
MAVs suffer, however, from stringent constraints on avionic payload, which requires highly miniaturized sensors, actuators and control systems (Viollet et al., 2008, Ahmad and Taib, 2003, Van Nieuwstadt and Morris, 1995).
Some of the aerial robots we developed thus far (FANIA and LORA) embedded only the optic flow sensors, the electromechanical actuators, and some internal stabilizing loops. The *visuo-motor control* system therefore operated off-board, yet real time experimentation was made possible by the joint use of *Simulink* (The Mathworks) and *Dspace* (Dspace) softwares. This permanent exchange between simulation (from the high level models) and experimental tests (from the robot) is appealing because it permits quick implementation of any new visuo-motor control systems onto a physical demonstrator, in addition to easy monitoring, and validation and tuning procedures. In this approach, the robot's behavior may be limited by the wire umbilical, however, unless a wireless link is established between the robot and the real-time board.
In our quest to achieve robot's complete - computational and energetic – autonomy, we now designed a complete digital system that integrates both optic flow sensors and robot's control systems within the same target: a 0.5 gram Field Programmable Gate Array (FPGA). An FPGA offers significantly more computational capabilities-per-gram than an embedded microprocessor or Digital-Signal Processor (DSP) because it supports custom, application-specific logic functions that may accelerate processing.
In the next sections, we describe our autopilot project based on an FPGA that meets three constraints: (i) the complete autonomy requirements, (ii) the computational requirement for real-time processing, and (iii) the requirement for light weight imposed by the very limited avionic payload (Chalimbaud and Berry, 2007).
Our project consists in the FPGA implementation of a *visuo-motor control system,* called LORA (LORA stands for *Lateral Optic flow Regulation Autopilot*), that was developed for a miniature hovercraft (section 3). The FPGA integration work was performed from a top-down design methodology using Intellectual Property (IP) cores and VHDL descriptions imported in the *Simulink* high-level graphical environment (The Mathworks) and the *System Generator* software interface (Xilinx) (section 2). The high-level *Simulink* blocks of the designed autopilot are then substituted for digital Xilinx blocks. Digital specifications of several functions, such as sampling time, fixed-point binary formats (section 4) and architectures (section 5), were defined from behavioral studies. Only models built from Xilinx blocks were translated into hardware logic devices using *System Generator*. Finally, the overall behavior of the integrated systems was analyzed using a hardware/software simulation based on both the *Simulink* environment (on a PC) and the FPGA (via a JTAG protocol). This co-simulation allowed us to analyze the hovercraft's behavior in various visual environments (section 6).

2. Design methodology for FPGA

2.1 FPGA general description

A field programmable gate array (FPGA) is a general-purpose integrated circuit that is programmed by the designer rather than the device manufacturer. Unlike an application-specific integrated circuit (ASIC), which can perform a similar function as in an electronic system, a FPGA can be reprogrammed, even after it has been deployed into a system. A FPGA is programmed by downloading a configuration program called a *bitstream* into static

on-chip random-access memory (RAM). Much like the object code for a microprocessor, this *bitstream* is the product of compilation tools that translate the high-level abstractions produced by a designer in an equivalent logic gate level.
A platform FPGA is developed from low-density to high-density designs that are based on IP cores and customized modules. These devices are user-programmable gate arrays with various configurable elements. The programmable device is comprised of I/O blocks and internal configurable logic blocks. Programmable I/O blocks provide the interface between package pins and the inside *configurable logic*.
The internal configurable logic includes elements organized in a regular array: *configurable logic blocks* (CLB) provide functional elements for combinatorial and synchronous logic, including basic storage elements; memory modules provide large storage elements of single or dual-port; Multiplier blocks integrating adder and accumulator (MAC); digital clock managers provide self-calibrating, fully digital solutions for clock distribution delay compensation, clock multiplication and division, coarse- and fine-grained clock phase shifting.
Several FPGAs also support the embedded system functionality such as high-speed serial transceivers, one or some hard embedded processors, Ethernet media-access control cores or others integrated high-functionality blocks.
A general routing matrix provides an array of routing switches between each component. Each programmable element is tied to a switch matrix, allowing multiple connections to the general routing matrix.
All programmable elements, including the routing resources, are controlled by values stored in memory cells. These values are loaded in the memory cells during configuration and can be reloaded to change the functions of the programmable elements.
FPGAs especially find applications in any area or algorithm that can use the massive parallelism offered by their architecture. They begin to take over larger and larger functions to the state where some are now marketed as full systems on chips (SOC). In this sense, they can satisfy the computational needs of real-time processing onboard autonomous MAVs.

2.2 FPGA design process

There are many design entry tools used for FPGA design. The easiest and most intuitive is the schematic entry. In this case, the required functionality is drawn using a set of library components. These one include the components primitives available on the FPGA as well as some higher level functions. Intellectual Property cores (IP) can be configured and placed in the schematic as a black box.
A trend in the digital hardware design world is the migration from graphical design entries to Hardware Description Languages (HDLs). They allow specifying the function of a circuit using a specific language such as VHDL or Verilog. These languages are specially made to describe the inherently parallel nature of digital circuits (behavioural and data flow models) and to wire different components together (structural models). In addition, HDLs are well adapted for designs with synchronous Finite State Machine (FSM) (Golson, 1994, Chambers, 1997). State machines impose a strict order and timing for operations making them similar to programs for CPUs.
However, HDL descriptions are difficult to design in the case of complex systems using digital signal processing functions or control applications. Indeed, the high level mathematical modelling tools are often used to validate a system model and to make a floating or fixed point model. Developments in the simulation capabilities of high-level

mathematical modeling tools have opened new design flow possibilities. These tools are includes in methodologies that help to hand complex design efficiently, minimize design time, eliminate many sources of errors, reduce the manpower required to complete the design and to produce optimal solutions. Nowadays, we can integrate complex system using IP cores and HDL descriptions imported in a high-level graphical environment. A software interface converts a fixed point model into hardware using traditional low level design implementation tools. Our design approach is based on this latter method.
In our project the high-level environment is *Matlab/Simulink* (The Mathworks), and the used interface tool is *System Generator for DSP* (Xilinx). *Matlab* is interactive software for doing numerical computations that simplifies the implementation of linear algebra routines. Powerful operations can be performed by utilizing the provided *Matlab* commands. *Simulink* is an additional toolbox of the *Matlab* software that provides a graphical environment for modeling, simulating and analyzing dynamic systems. *System Generator for DSP* toolbox was specifically designed to allow the fast development of complex system requiring powerful digital signal operations. It is presented in the form of a 'toolbox' added to the *Simulink* graphic environment, which allows the interfacing of certain functions fulfilled with the others 'toolbox' dedicated to *Simulink* (Turney, 1999, Ownby, 2003).

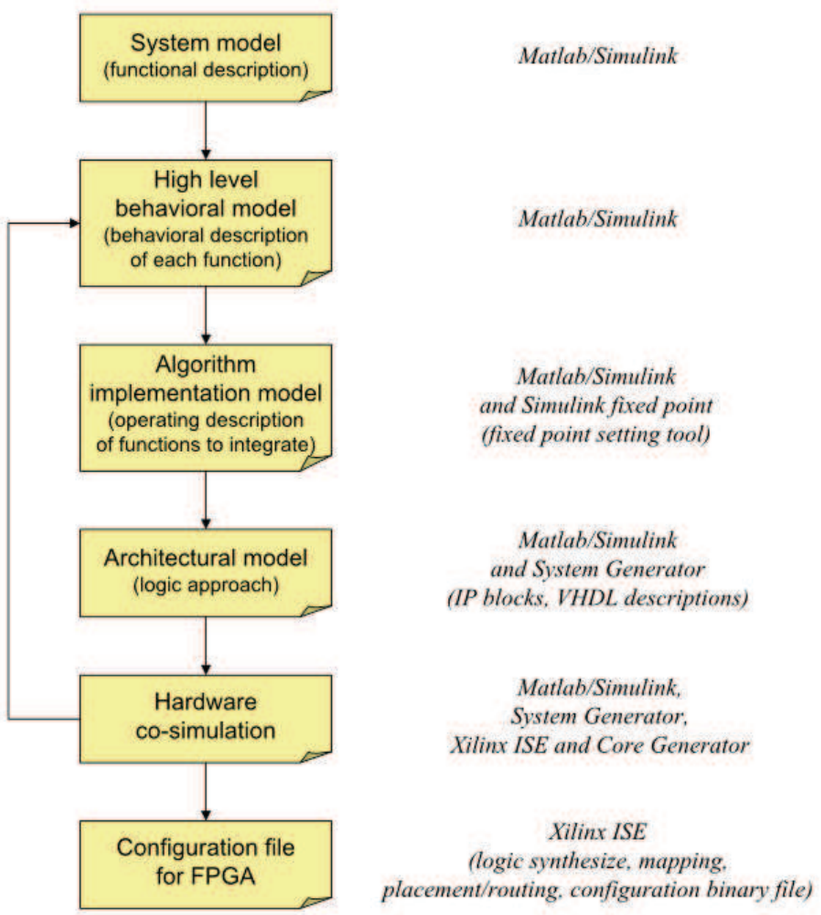


Figure 3. Top-down design methodology with a view to designing, simulating and implementing 'visuo-motor control loops' in FPGA (from Aubépart and Franceschini, 2007)

The design flow is presented in Figure 3. This methodology requires design various stages as well as the linking framework. This top-down methodology simplifies the integration problems by using detailed descriptions (Browny et al., 1997; Aubépart & Franceschini, 2005, 2007, Murthy et al., 2008).

Firstly, we start to create a '*system model*' as well as one or more simulated environment configurations for validating its principle. In this state, a functional approach involves dividing the system into elementary function blocks. The elements used in constructing the '*system model*' are all capable of operating on real (double precision, floating point) or integer (quantized, fixed point binary) data types. When the model is first entered, simulation is typically performed using floating data types to verify that its theoretical performance is as desired.

Secondly, we define the '*high-level behavioral model*' which describes the computation sequences and timing data (sampling frequency, delays, etc.). For example, mathematical operators that occur within an algebraic feedback loop may have an associated delay to avoid the system instability. The fixed time step, insert delays and/or rate changes need to be also considered to ensure a stability of a system based on a feedback loop.

Thirdly, an '*algorithm implementation model*' must be defined for each function to integrate. This model identifies the data type and the procedures used in each function. It takes account some factors, such as the binary format, in order to optimize the digital calculations. The internal data types are then converted to the bit true representations that will be used in the hardware implementation, and the model is re-simulated to verify its performance with quantized coefficient values and limited data bit widths, which can lead to overflow, saturation and scaling problems.

Fourthly, the hardware constraints are study in the '*architectural model*' that defines one or several implementation architectures for each function to integrate. They are replaced by IP blocks available in the *System Generator* toolbox. We can also define black boxes, such as VHDL descriptions, which can be incorporated into the model and the elaboration process. The importation of VHDL descriptions will be limited to complex synchronous descriptions, such as Finite State Machines.

Fifthly, the final verification will be completed by implementing the hardware co-simulation of the *System Generator* '*architectural models*'. The co-simulation process uses *Xilinx ISE* and *core generator* to synthesize and generate and FPGA programming bit file. A new *Simulink* library was created containing the hardware co-simulation blocks. These blocks were copied into the *Simulink* project file for replacing all the *System Generator* '*architectural models*'. The port names, types and rates are matching the original design. The hardware implementation is then executed by connecting the FPGA board to the computer and using a standard JTAG connection. When the simulation is run, stimuli are send to FPGA and output signals are receive by *Matlab/Simulink* environment, closing the loop.

Finally, the designer can invoke the *netlister* and test-bench generator available from *System Generator*. The *netlister* extracts a hierarchical VHDL representation of the model's structure annotated with all element parameters and signal data types that will integrate into FPGA from traditional low level tools include in *Xilinx ISE*: logic synthesize, mapping, placement and routing, generate programming file.

3. LORA III autopilot

In this part, we present the 'system model' and the implementation of a full control system using *System Generator* toolbox. This study integrate a complete *visuo-motor control system*, called LORA III (LORA stands for Lateral Optic flow Regulation Autopilot, Mark 3), which was designed for a particular kind of aerial vehicle: a fully actuated miniature hovercraft that is able to "fly" at a few millimetres above the ground and to control both its speed and its distance from the walls – without any measurements of speed and distance (Serres et al., 2008).

3.1 Robot position in travel environment

The hovercraft is simulated to travel through an unknown textured corridor at a ground speed vector $\vec{V}$ over a flat surface, Figure 4. The walls are lined with vertical stripes with random spatial frequency and contrast that mimic a richly textured environment (Iida, 2001, Ruffier & Franceschini, 2005). The corridors are of two types: right or tapering.

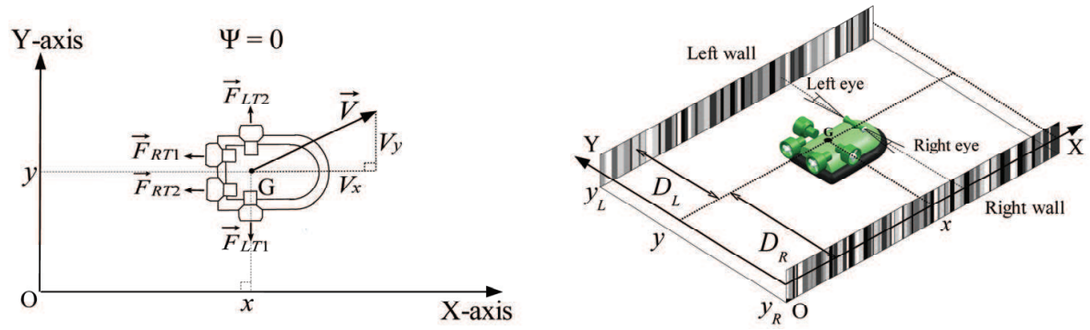


Figure 4. Miniature hovercraft moving through an unknown textured corridor: the groundspeed vector $\vec{V}$ is projected onto the corridor-fixed coordinate frame. Four thrusters (two rear thrusters and two lateral thrusters) allow the hovercraft to be fully actuated in the plane. (Adapted from Serres et al., 2008)

The robot's position (x, y) is computed in including side forward speed V_x and the side speed V_y. The hovercraft is fully actuated because in addition to the pair of rear thrusters providing forward motion (surge axis) and heading control (yaw axis), the vehicle is equipped with a pair of lateral thrusters generating independent side-slip motion (sway). The angle ψ defines the hovercraft's yaw angle with respect to the corridor axis, which is kept at a reference value equal to zero (ψ = 0). In this study, the hovercraft's heading ψ is assumed to be stabilized along the X-axis of the corridor.

The hovercraft's motion is defined by dynamic equations involving the forward thrust ($F_{Fwd} = F_{RT1} + F_{RT2}$) produced by the rear thrusters (left: RT1, right: RT2) and the lateral thrust ($F_{Side} = F_{LT2} - F_{LT1}$) produced by the lateral thrusters (left: LT1, right: LT2). In the simulations, the maximum forward speed is 2m/s and the maximum side speed is 0.5m/s. At such low speeds, the drag-versus-speed function can be linearized. The following equations referred to the center of gravity G define the dynamics of the simulated hovercraft (Figure 5):

$$\begin{aligned} m.\frac{dV_x}{dt} + \xi_x.V_x &= F_{RT1} + F_{RT2} = K_T.(U_{RT1} + U_{RT2}) \\ m.\frac{dV_y}{dt} + \xi_y.V_y &= F_{LT2} - F_{LT1} = K_T.(U_{LT2} - U_{LT1}) \end{aligned} \quad (1)$$

Where m = 0,73 kg is the total mass of the hovercraft, and ξ_x and ξ_y are translational viscous friction coefficients along the X-axis and Y-axis, respectively. K_T (0.10 N/V) is a simple gain that relates the thrust to the applied voltage: U_{RT1} and U_{RT2} are the forward control signals received by the rear thrusters, U_{LT2} and U_{LT1} are the side control signals received by the lateral thrusters.

3.2 Dual optic-flow regulator

The system addresses both issues of automatic speed control and side wall avoidance of the miniature hovercraft simultaneously. LORA is a dual optic flow regulator that consists of two interdependent control loops: a forward control loop and a side control loop. Figure 5 shows the block diagram involving multiple processing stages. This scheme is composed of two parts. In first, all functions that we would integrate into FPGA (blue, green and red functions). Secondly, hovercraft dynamics, lens/photoreceptors system and visual environments simulate the trajectories resulting from the LORA dual regulator scheme (Cyan and cyan hatched functions). These '*high level behavioural models*' will be used when digital, timing and architecture specifications will be defined for the functions to integrate.

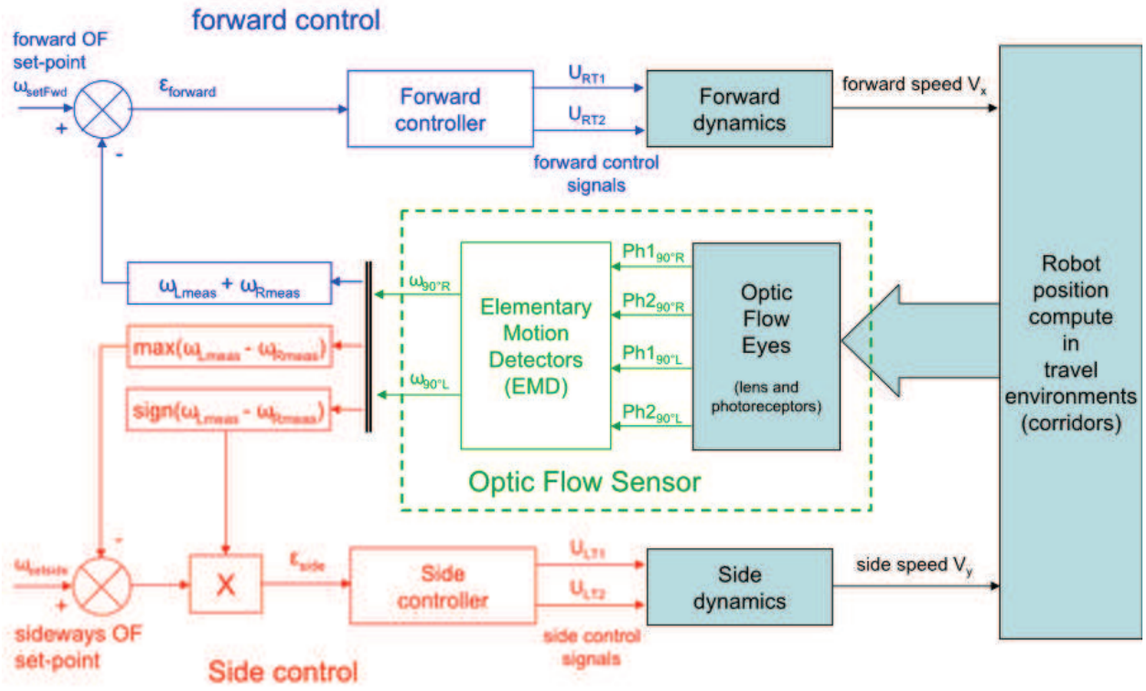


Figure 5. LORA III functional block diagram. LORA III autopilot is based on two interdependent visual feedback loops working in parallel with their own optic flow set-point (the *forward control system is the upper one, and the side control* system is the bottom one). *Optic flow sensors measure the right and left* optic flows in accordance with the hovercraft's speed and position in the environment (adapted from Serres et al., 2008a)

In Serres et al. (2008), the full control system is described in detail. We summarize here the principal aspects of them. That the hovercraft is fully actuated means that each groundspeed component V_x and V_y can be controlled independently. LORA III regulates (i.e., maintains constant) the lateral OF by side and forward controls, according to the following principles:

(i) The first *lateral OF regulator* adjusts the air vehicle's lateral thrust (which determines the lateral speed V_y, i.e., the sway speed) so as to keep the lateral optic flow equal to the *sideways OF set-point*. The outcome is that the distance to the wall becomes proportional to the vehicle's forward speed V_x which is defined in (ii): the faster the air vehicle travels, the further away from the walls it will be. The clearance from the walls will depend directly on the *sideways OF set-point*.
(ii) The second *lateral OF regulator* adjusts the air vehicle's forward thrust (which determines the forward speed V_x, i.e., the surge speed) so as to maintain the sum of the two (right and left) optic flows equal to the *forward OF set-point*. The outcome is that the air-vehicle travels all the faster as the environment is less cluttered. The forward speed attained by the vehicle will depend directly on the *forward OF set-point and will become proportional to the local width of the corridor*.
The first lateral optic flow regulator is based on a feedback signal that takes into account the left or right optic flow measured. The feedback is simply the larger of the two optic flows measured (left or right): $\max(\omega_{Lmeas}, \omega_{Rmeas})$: it corresponds to the nearest wall: $\min(D_L, D_R)$.
This optic flow regulator was designed to keep the lateral optic flow constantly equal to the side optic flow set-point $\omega_{SetSide}$. The hovercraft then reacts to any deviation in the lateral optic flow (left or right) from $\omega_{SetSide}$ by adjusting its lateral thrust, which determines the hovercraft's side speed V_y: this eventually leads to a change in the distance to the left (D_L) or right (D_R) wall. A sign function automatically selects the wall to be followed, and a maximum criterion is used to select the higher optic flow value measured between ω_{Rmeas} and ω_{Lmeas}. This value is then compared with the sideways optic flow set-point $\omega_{SetSide}$. The error signal ε_{side} feeding the side controller is therefore calculated as follows:

$$\varepsilon_{side} = \text{sign}\left(\omega_{Lmeas} - \omega_{Rmeas}\right) \times \left(\omega_{SetSide} - \max\left(\omega_{Lmeas}, \omega_{Rmeas}\right)\right) \tag{2}$$

The identified transfer function of the side dynamics $G_y(s)$ relating the hovercraft's ordinate *y* to the control signal approximates a first-order low-pass filter (with a time constant of 0.5s) in series with an integrator:

$$G_y(s) = \frac{Y(s)}{(U_{LT1} - U_{LT2})(s)} = \frac{1}{s} \times \frac{\frac{K_T}{\xi_y}}{1 + \frac{m}{\xi_y} s} = \frac{1}{s} \times \frac{0.1}{1 + 0.5s} \tag{3}$$

A lead controller $C_y(s)$ was introduced into this feedback loop to increase the damping, thus improving the stability and enhancing the response dynamics. The lead controller $C_y(s)$ (Eq.3) is tuned to reach a phase margin of 45° and a crossover frequency of 4rad/s (0.64Hz):

$$C_y(s) = \frac{(U_{LT1} - U_{LT2})(s)}{\varepsilon_{side}(s)} = 10 \times \frac{1 + 1.5s}{1 + 0.5s} \tag{4}$$

The second optic flow regulator is the forward control system. It is intended to keep the sum of the two lateral optic flows measured ($\omega_{Rmeas}+\omega_{Lmeas}$) constant and equal to a *forward optic flow set-point* ω_{SetFwd} by adjusting the forward thrust, which will determine the hovercraft's

forward speed V_x. At a given corridor width, any increase in the sum of the two lateral optic flows is assumed here to result from the hovercraft's acceleration. This control scheme thus automatically ensures a 'safe forward speed' that is commensurate with the local corridor width. The sum of the two optic flows measured is compared with a *forward OF set-point* ω_{SetFwd}, and the error signal ε_{Fwd} (the input to the forward controller) is calculated as follows:

$$\varepsilon_{Fwd} = \omega_{SetFwd} - (\omega_{Rmeas} + \omega_{Lmeas}) \tag{5}$$

The model $G_{Vx}(s)$ for the dynamics of our hovercraft is described by a first order low-pass filter with a time constant of 0.5s:

$$G_{Vx}(s) = \frac{V_x(s)}{(U_{RT1} + U_{RT2})(s)} = \frac{\frac{K_T}{\xi_x}}{1 + \frac{m}{\xi_x}s} = \frac{0.1}{1 + 0.5s} \tag{6}$$

A proportional-integral (PI) controller $C_{Vx}(s)$ (Eq. 7) is tuned to cancel the dominant (aeromechanical) pole of the hovercraft and to reduce the forward time constant computed in the closed-loop by a factor of 1.57. The integral action is introduced to cancel the steady state error:

$$C_{Vx}(s) = \frac{(U_{RT1} + U_{RT2})(s)}{\varepsilon_{Fwd}(s)} = 10 \times \frac{1 + 0.5s}{s} \tag{7}$$

3.3 Bio-inspired visual system

The visual system of the hovercraft consists of two optic flow sensors. Each optic flow sensor is designed by a lens/photoreceptor assembly (the eye) including at least two photoreceptors (i.e. two pixels) driving an Elementary Motion Detector (EMD) circuit.

Each eye consists of two photoreceptors mounted slightly defocused behind a lens, which creates a bell-shaped Angular Sensitivity Function (ASF) for each of them (Hardie, 1985). The ASF, which is often modelled in the form of a truncated Gaussian curve (Netter & Franceschini, 2002) and characterized by the 'acceptance angle' $\Delta\rho$=4° (i.e. the angular width at half height). The ASF plays an important role in the visual processing chain, because it serves as an effective low pass anti-aliasing spatial filter. The visual axes of each pixel are separated by an inter-receptor angle $\Delta\varphi$ = 4°.

The EMD principle was originally based on the results of experiments in which a combined electrophysiological and micro-optical approach was used. The activity of a large field motion detecting neuron in the housefly's eye was recorded with a microelectrode while applying optical microstimuli to a single pair of photoreceptor cells located behind a single facet (Franceschini, 1985, Franceschini et al., 1989). Based on the results of these experiments, a principle was drawn up for designing an artificial EMD capable of measuring the angular speed ω of a contrasting object (Franceschini et al., 1986, Blanes, 1986).

Thus, in each of EMDs, the lens/photoreceptor combination transforms the motion of a contrasting object into two successive photoreceptor signals separated by a delay Δt:

$$\Delta t = \frac{\Delta\varphi}{\omega} \tag{8}$$

Where $\Delta\varphi$ is the inter-receptor angle and ω is the relative angular speed (the optic flow). An electronic device based on some linear and nonlinear functions estimates the angular speed ω_{EMD}:

$$\omega_{EMD} = e^{-\left(\frac{\Delta t}{\tau}\right)} \Leftrightarrow K.\omega = \frac{K.\Delta\varphi}{\Delta t} = \frac{K'}{\Delta t} \tag{9}$$

Our original EMD functional scheme (Franceschini et al., 1986, Blanes 1986, Aubépart & Franceschini, 2007) consists of five processing steps giving ω_{EMD} (Figure 6):

1. A first-order high-pass temporal filter (f_c = 20Hz) produces a transient response whenever a contrasting border crosses the photoreceptors' visual field. This filter enhances the contrast information while eliminating the DC components of the photoreceptor signals.
2. A higher order low-pass temporal filter (f_c = 30Hz) attenuates any high frequency noise, as well as any interferences brought about by the artificial indoor lighting (100Hz) used.
3. A thresholding device/step normalizes the signals in each channel.
4. A time delay circuit is triggered by one channel and stopped by the neighboring channel. This function measures the time Δt elapsing between similar transitions occurring in two adjacent photoreceptors.
5. A converter translates the delay Δt measured into a monotonic function that will approximate the angular speed ω_{EMD}. A simple inverse exponential function makes for a relatively large dynamic range (eq. 9).

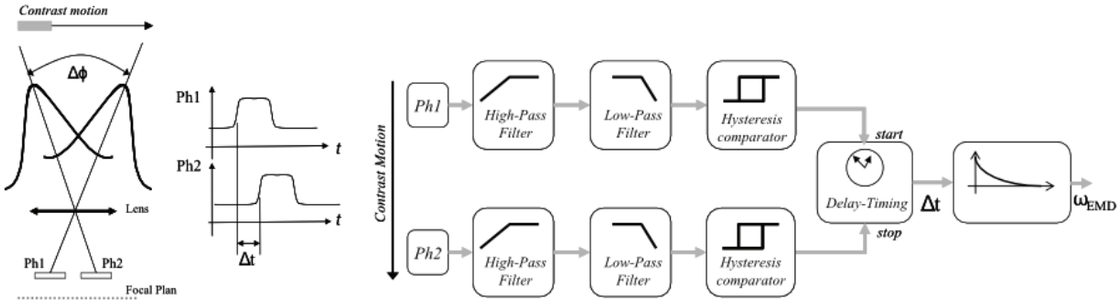


Figure 6. Principle scheme of Elementary Motion Detector (EMD). (Aubépart & Franceschini, 2007)

4. Implementation of visual feedback loops

4.1 Sampling time consideration

In aerial robotic applications, the sampling time must comply with the requirements imposed by the digital control system so that the MAV can be controlled throughout its safe flight envelope. The maximum sampling time T_{SMAX} will depend on the minimum delay Δt_{min} encountered by the robot's EMDs during the fastest manoeuvres in the most critical applications. Indeed, in an extreme case the smallest delay Δt_{min} which could be determined by the EMD will be that corresponding to one only counter clock period, whose clock signal corresponds to the sampling time.

One example of a fast maneuver is automatic wall-following, which is performed by measuring the side optic flow in the side direction. When the eye-bearing hovercraft is in pure translation at speed V_x and follows a textured wall at distance D, the image of the wall slips at an angular speed ω that depends on both V_x and D:

$$\omega = \frac{V_x}{D} = \frac{\Delta\varphi}{\Delta t} \tag{10}$$

If we take an extreme case where the hovercraft is allowed to follow the wall at the minimum distance D = 0.05m at the maximum speed V_x = 2m/s, equations 8 and 10 show that the 90° oriented EMD onboard the hovercraft, with its inter-receptor angle $\Delta\varphi$ = 4°, will be subject to a minimum delay $\Delta t_{min} \approx 1.8$ms. Accordingly, the sampling frequency f_{Smin} will have to be set at values of at least 500Hz. When considering a MAV flying over an unknown terrain, we selected a minimum sampling frequency of 1kHz (Aubépart & Franceschini, 2007).

The maximum sampling frequency f_{SMAX} is less constraining to choose. It must be in keeping with the timing specifications to which the lens/photoreceptors devices are subject, especially in the case photoreceptors using the current-integrator mode (Kramer et al., 1997). On the other hand, the maximum sampling frequency f_{SMAX}, is limited by the lower end of the illuminance range over which the sensor is intended to operate, because at low illuminance levels, the integration of the photoreceptor signal takes a relatively long time and the sampling procedure will have to wait for this integration process to be completed (Aubépart & Franceschini, 2007). Taking the range [100Lux–2000Lux] to be a reasonable working illuminance range for the hovercraft, this gives f_S = 2.5kHz, which we call the 'nominal sampling frequency' (Aubépart & Franceschini, 2005). At twice this sampling frequency (5kHz), the hovercraft would still operate efficiently in the [200Lux – 2000Lux] range, but it would then be difficult for it to detect low contrasts under artificial indoor lighting conditions. In addition, we avoided CMOS cameras equipped with digital outputs, which have not high frame rates, which still need to scan every pixel internally (Yamada et al., 2003, Zufferey et al., 2003).

As regards the forward and side controllers or others functions ('sign', 'add' and 'max' functions) present in the visual feedback loops, the sampling time should be adapted to the maximum frequency of the processed signals. Generally, the highest frequency corresponds to the smallest time-constant of control loops. In our loops, it is equal to 0.5-second that corresponds to a cut-off pulsation of 2rad/s. In discrete time we may consider that a sampling frequency 100 times higher than the system cut-off frequency would give results similar to continuous time. An equivalent sampling frequency of about 32 Hz could be sufficient. Since this value is in the lower part of the minimal value necessary to EMDs, we chose 2.5kHz as the nominal sampling.

4.2 Digital specifications

In feedback control loops, the digital specifications were mainly defined for EMDs design (i) and controllers design (ii). The secondary functions, such as 'sign', 'add' and 'max' functions, are easily adapted to the digital choices (binary format, implementation, etc.) without modifying the control loops operation.

In EMDs (i), the digital specifications were found during the filter design. Due to the low values of the high-pass and low-pass filter corner frequencies (f_{CHP} = 20Hz, f_{CLP} = 30Hz) in comparison with the sampling frequency (f_S = 2,5kHz), it was not possible to obtain a digital band-pass filter meeting the Bode specifications. The high-pass filter section and low-pass filter section were therefore designed separately and cascaded.
Infinite Impulse Response (IIR) filters were synthesized (see eq. 5 below) because they require far fewer coefficients than Finite Impulse Response (FIR) filters, given the low cut-off frequencies and short sampling times involved:

$$y(n) = \sum_{i=1}^{n} b(i) \cdot x(i) - \sum_{i=1}^{n-1} a(i) \cdot y(i) \tag{11}$$

A transposed Direct-Form II structure was used because this structure reduces the number of delay-cells and decreases the quantization errors. Ripples on the low-pass filter temporal response were prevented by using a 4th-order Butterworth Filter, the phase of which was linearized over the frequency range of interest. The filters require 17 coefficients in all (4 coefficients for the 1st-order high-pass section, 12 for the 4th-order low-pass section, and 1 for the adjustment between the two filters). Three Direct-Form II filters suffice in fact to perform all the filtering, including that carried out by the two cascaded 2nd order low-pass filters.
A specific binary format was developed and used to prevent offset and stabilization problems. A two-complement fixed-point binary format, denoted [s,m_I,m_D], was defined. The bit number of integer parts, m_I, and the decimal part, m_D, were defined so as to ensure maximum accuracy and to eliminate overflow from the filter calculations. Based on the results of a study carried out with *Filter Design and Analysis* and *Fixed-point Blockset* of the Mathworks tools, 6 bits were selected for the integer part m_I and 29 bits for the decimal part m_D. The large m_D bit number is due to the low value required to make the coefficients in the low-pass filter section comply with a Bode template characterized by a low cut-off frequency at high sampling frequencies.
Other digital specifications were defined in EMDs as regards the bit number of the counter output giving the delay time Δt, and the inverse exponential function giving the angular speed ω_{EMD}.
The delay time Δt is measured in terms of a count number at a given clock period. The minimum delay to be measured determines the minimum clock period (400µs for f_S = 2.5kHz). The maximum delay to be measured is taken to be ~100ms, which is compatible with the wide range of angular speed values encountered by the hovercraft eyes (eq.4): ω ~40°/s to ~10000°/s, for $\Delta\varphi$ = 4°. Using an 8-bit counter at f_S = 2.5kHz gives an delay of 102.4ms.
The measured angular speed ω is a hyperbolic function of Δt (eq. 8), but we used a function that decreases more slowly: an inverse exponential function with a time constant τ = 30ms. A Look-Up Table (LUT) was used to convert the delay Δt into an output that decreases monotonically (exponentially) with the delay and therefore approximately reflects the angular speed ω_{EMD} (eq. 9). The Look-Up Table features an 8-bit input resolution (at f_S = 2.5kHz) and a 12-bit output resolution for memorizing the results of the conversion.
The digital correctors design (ii) is easy in using 'c2d' Matlab function (discretize continuous-time system). The correctors' continuous-time models have been converted to

discrete-time models with sampling time T_S. The discretization method used was the bilinear Tustin approximation. Table 1 presents the discretized polynomial transfer functions.

Correctors	*Continous-time*	*Discrete-time*
side	$C_y(s) = 10 \times \frac{1+1.5s}{1+0.5s}$	$C_y(z) = \frac{29.99z - 29.98}{z - 0.9992}$
forward	$C_{Vx}(s) = 10 \times \frac{1+0.5s}{s}$	$C_{Vx}(z) = \frac{5.002z + 4.998}{z - 1}$

Table 1. Correctors' discretization using Tustin bilinear approximation with f_S = 2.5kHz

As during EMD design, a specific binary format was defined to avoid offset and stability problems. Once again, we used the '*Filter Design and Analysis*' tool as well as the '*Fixed-point Blockset*' tool. A first approximation gave two signed fixed-point binary formats: 9 bit and 8 bit for the integer part m_I of the forward and side correctors, respectively, 10 bit for their decimal parts m_D. However, simulations of the full feedback control system showed that accuracy was too low. The decimal parts m_D were then increased to 13 bit for best accuracy in computation.

5. Architecture

5.1 Optic flow sensor

Figure 7 shows the "EMD *architecture*". This architecture has several important features, such as the optimization of the digital filters, the simplicity of design owing to the use of Intellectual Property (IP) cores, and the added flexibility in the circuit design. Special care was taken to restrict the space taken by the digital filter implementation. Instead of implementing eight IIR digital filters in parallel - which would each require a high number of mathematical operators (adders, subtracters and multipliers) - a single structure called the "*Filter Compute Unit*" was developed, with which high speed sequential processing can be performed, as shown in figure 8. This unit consists of only one multiplier, one adder, one Read Only Memory (ROM), one Random Access Memory (RAM), two multiplexers, three registers and two binary transformation functions. The ROM contains the 17 filter coefficients obtained at a sampling frequency f_S = 2.5kHz. The RAM is used to store the intermediate values computed. The multiplexers minimize the number of operators.

In this unit, each photoreceptor signal is processed during the sampling time T_S. A Finite State Machine (FSM) was written in VHDL language and imported into the *Simulink* environment. The FSM specifies the filtering sequence for each photoreceptor channel. Even though this solution somehow complicates the design (due to the mixing of VHDL description and System Generator IP blocks), it has the advantage to minimize the number of logical gates necessary for integration in the FPGA.

Once the filtering process is completed, the delay Δt between the excitations of two neighboring photoreceptors starts being measured. A comparator determines the instant at which the band-pass filtered signal from each photoreceptor channel reaches the threshold value. The resulting logical signal is used to trigger the measurement of the delay Δt corresponding to each of the two eyes (an eye consists of a lens and only two photoreceptors). Specifically, the logical signal delivered by a photoreceptor starts a counter that will be stopped by the logical signal delivered by the neighboring photoreceptor.

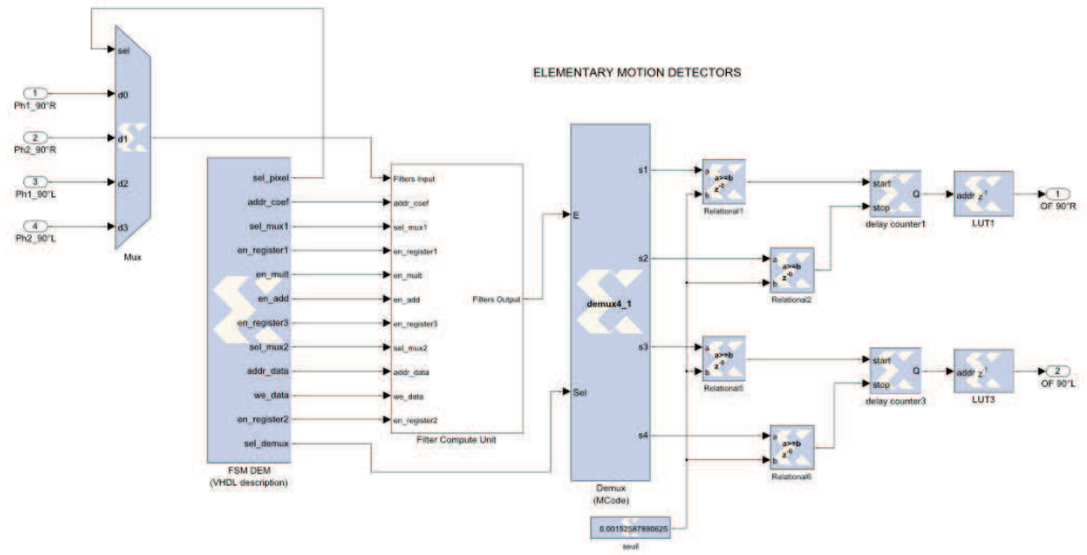


Figure 7. Elementary Motion Detector Architecture

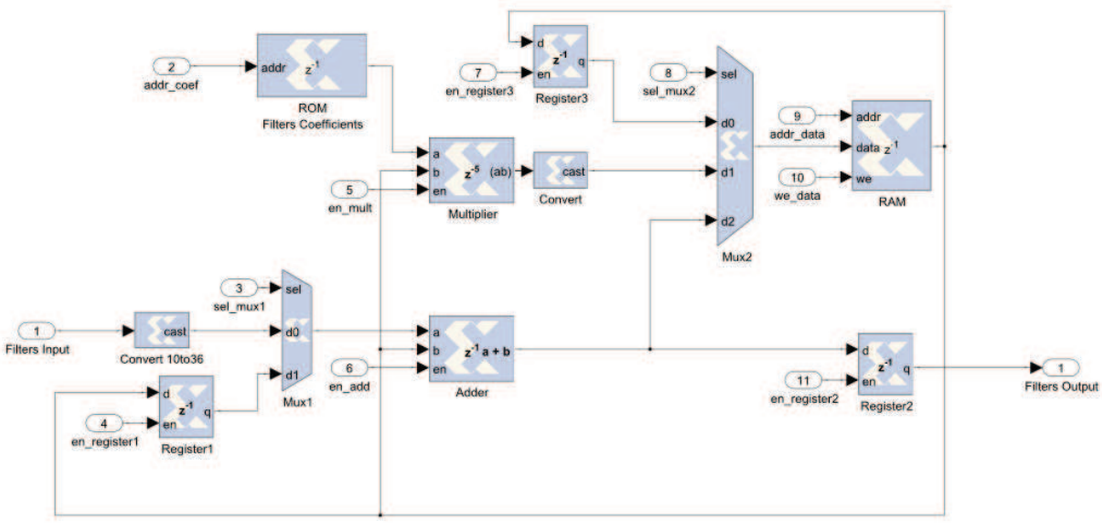


Figure 8. "Filter Compute Unit" architecture. The Filters' input is at port Nb 1. Others inputs (from 2 to 11) are control signals from a Finite State Machine that specifies the sequence of the filtering process for each photoreceptor channel

The final piece of the architecture is the inverse exponential function with a time constant τ = 30ms, which allows for a wide range of delays ranging up to 102.4ms. This component was implemented in the form of Look-Up Tables (ROM) to facilitate the conversion of each delay data into an estimated angular speed ω_{EMD} (i.e., the optic flow).

5.2 Forward and Side correctors

The forward and side controllers (see Figure 5) have discretized polynomial transfer functions that can be designed as first-order IIR filters. For their realization, we chose the transposed-Direct Form-II architecture. *System generator* IP blocks allow for graphical

integration of IIR filters because they use only multipliers, registers and adders. Using the Xilinx Multiplier Block called "Mult" (instead of the "CMult Xilinx Constant Multiplier") allowed this operation to be integrated via the multipliers embedded in the FPGA.

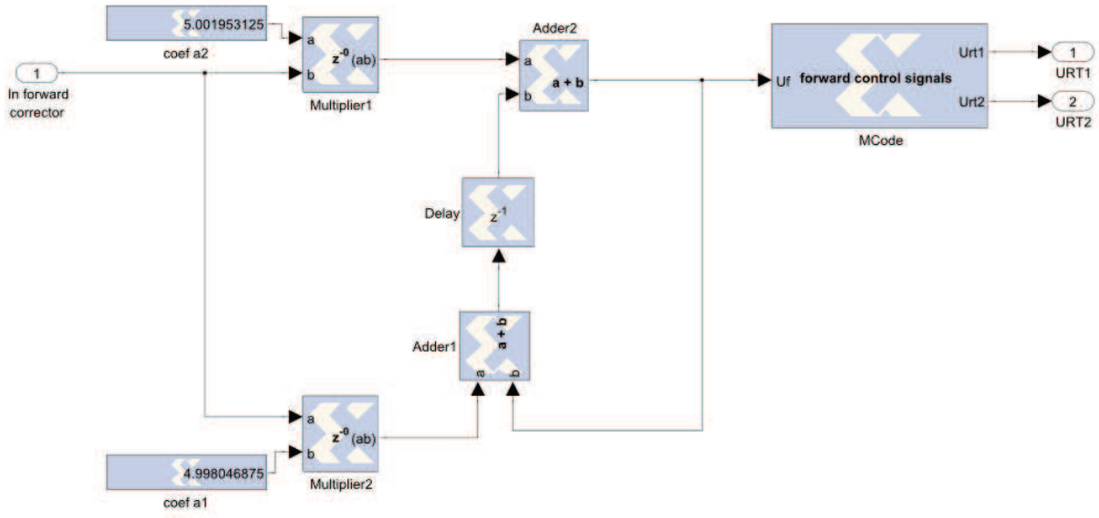


Figure 9. Transposed-Direct-Form-II implementation for the *forward controller* (input to the left, outputs to the right)

Figure 9 shows the forward controller, in which only two multipliers are implemented because the denominator coefficients are equal to one (see Table 1). A Xilinx "MCode" block allows the two forward control signals (U_{RT1} + U_{RT2}) to be determined, for driving the two rear thrusters RT1 and RT2. The Xilinx "MCode" block is a container that executes a user-supplied MATLAB function within *Simulink*. A parameter of this block specifies the M-code function name. The block executes the M-code and calculates the block outputs during the *Simulink* simulation. When hardware is generated, the same code is translated into equivalent behavioral VHDL in a straightforward manner.

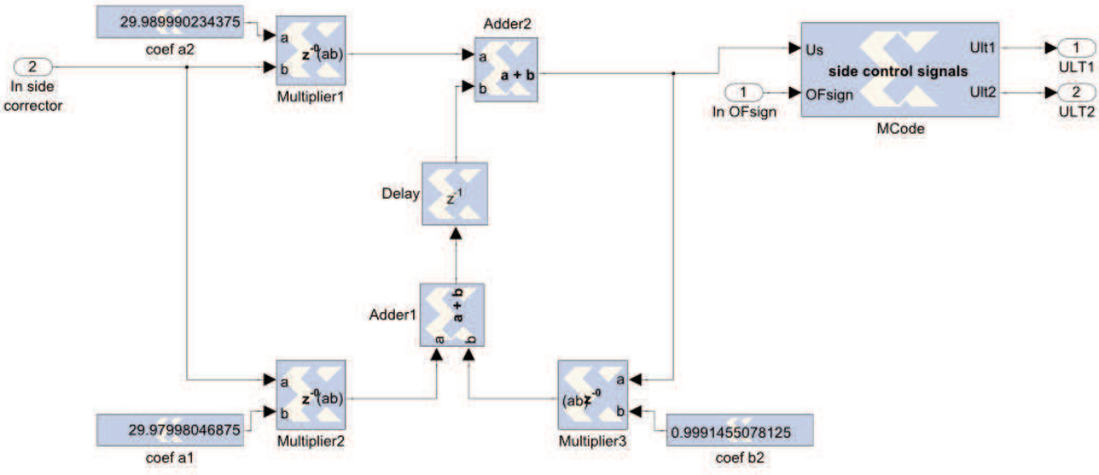


Figure 10. Transposed-Direct-Form-II implementation for the *side controller* (input to the left, outputs to the right)

The *side controller* was designed in the same way as the *forward controller*, figure 10. An additional multiplier was needed to perform calculation with the denominator coefficient b2. The two side control signals (U_{LT1} - U_{LT2}) controlling the two lateral thrusters LT1 and LT2 were again obtained by a Xilinx "MCode" block in which the sign of the difference between the right and left lateral optic flows measured was used.

5.3 Complete visuo-motor control system

The complete control system based on *System Generator* is presented in Figure 11. The secondary functions such as the determination of the larger value (OF max) and sign between right and left optic flows measured are easily realized with Xilinx "MCode" blocks. An adder and two subtracters are put in to close the loops. Xilinx "Gateway Out" blocks and "Gateway In" blocks limit the I/O from the Xilinx portion of the user's *Simulink* design. "Gateway Out" blocks convert the *System Generator* fixed point data type into *Simulink* double format while "Gateway In" blocks convert *Simulink* integer, double and fixed point data types into the *System Generator* fixed point type. Each block will define a top-level I/O port in the HDL design generated by *System Generator*.
Forward dynamics, side dynamics and "robot position compute" are defined by high level *Simulink* models linked to Matlab files that set several constants before simulation. "Lens/photoreceptors model" is designed with S-functions. An S-function is a computer language description of a *Simulink* block and is used to add our own blocks to *Simulink* models. Generally, such functions make it possible to accelerate simulations.

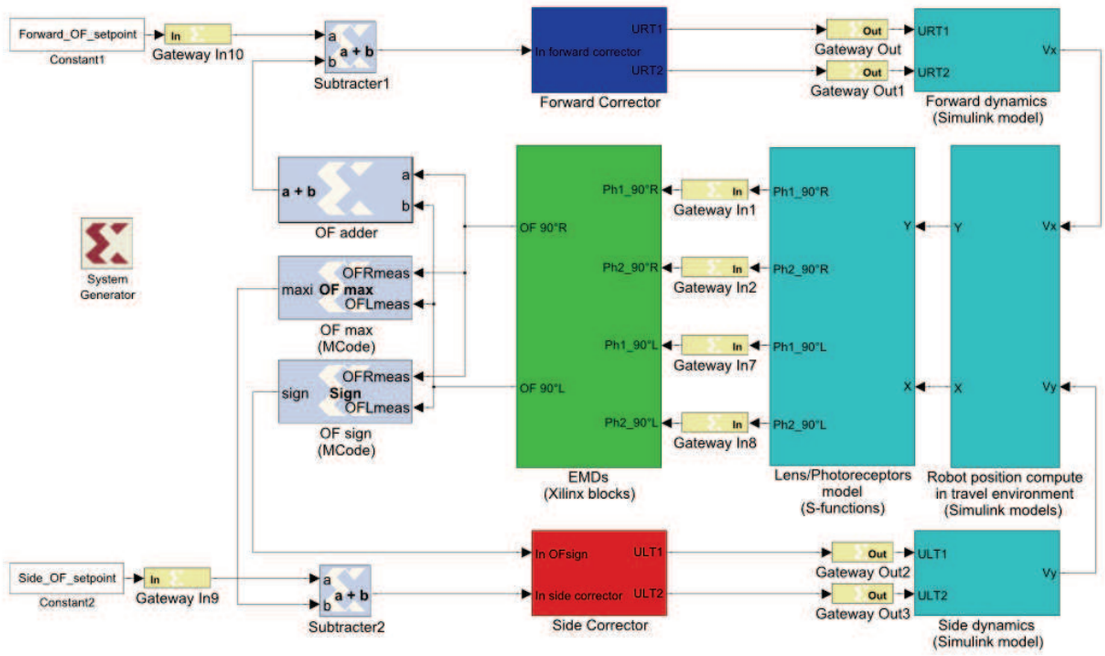


Figure 11. Complete control system based on *System Generator*. Xilinx "Gateway Out" blocks and "Gateway In" blocks limit the I/O from the Xilinx portion of the user's *Simulink* design

The *System Generator* block provides for control of system and simulation parameters, and is used to execute the code generator. Every *Simulink* model containing any element from the Xilinx Blockset must contain at least one *System Generator* block. Once a *System Generator*

block is added to a model, it is possible to specify how code generation and simulation should be handled.

6. Hardware co-simulation

The final tests were achieved by making a hardware co-simulation of the system. The various functions (the EMDs and the two complete visuo-motor control loops) were implemented on a small Virtex-4 FPGA (type XC4FX12, size 17mm X 17mm, mass 0.5gram) whose digital processing capacities are largely sufficient. We designed a specific electronic board, figure 12, embedding the FPGA to validate the various digital properties of the whole system in co-simulation. This board was realized with an aim to install it on-board the miniature hovercraft and therefore also included the required interface components (ADC, DAC, etc.). The complete board weighs only 17.3grams and measures 90mm x 50mm and it is well suited to an embedded technological solution.

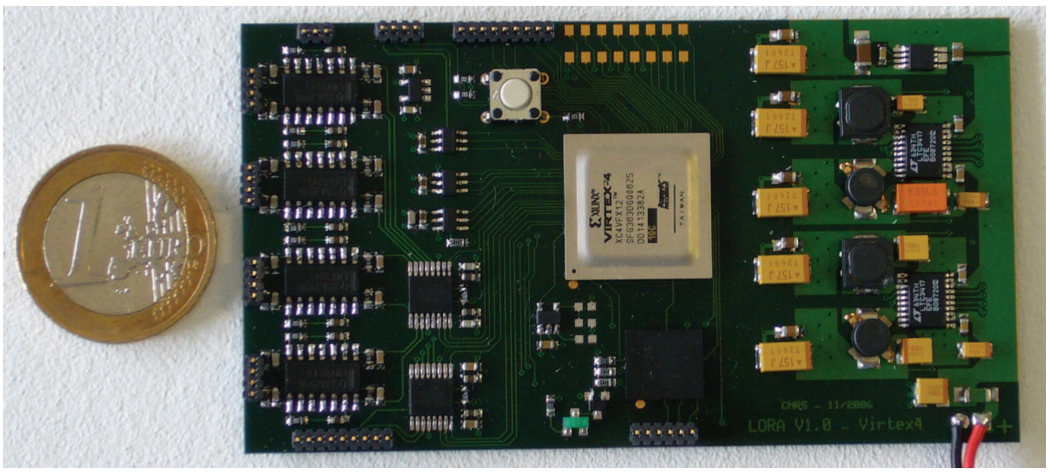


Figure 12. Specific electronic board based on a small FPGA Virtex-4: (i) right hand side: components for the power supply; (ii) middle part: FPGA XC4FX12 (top) and the Flash-memory to configure it (bottom); (iii) left hand side: electronic front-end related to the photosensors and several Analog-Digital and Digital-Analog Converters

The FPGA power consumption was evaluated using Xilinx "XPower" tool. The consumption is estimated at 149mW for a 2.5 kHz sampling frequency of the visuo-motor control loops and for the FPGA running at a clock frequency of 4MHz. The total power consumption of the electronic board, figure 12, is estimated at ~500mW

6.2 Hardware co-simulation results

A *Simulink* library was created from *System Generator* and copied into the *Simulink* project file replacing all the Xilinx System Generator blocks (i.e. between "Gateway In" and "Gateway Out" blocks, figure 11). The simulated robot is equipped with two lateral eyes oriented at ±90° to the walls (the inter-receptor angle is $\Delta\varphi$=4° and the acceptance angle is $\Delta\rho$=4°). At 2.5kHz sampling frequency, the computing temporal step is δt = 400µs and the simulation spatial accuracy is $\delta\theta$ = 0.005°. The two OF set-points were chosen according to the results of behavioural studies on honeybees that were video-filmed when flying through

a straight corridor (Serres et al., 2008b). The forward OF set-point was set to $\omega_{setFwd} = 314°/s$ and the side OF set-point was set to $\omega_{setSide} = 238°/s$.

6.2.1 Automatic speed control and lateral positioning in a straight corridor

The simulated visual environment is a 3-meter long, 1-meter wide straight corridor with textured walls. The right and left walls are lined with a random pattern of various grey vertical stripes covering a 1-decade contrast range (from 4% to 38%) and a 1.1-decade angular frequency range (from 0.068 c/° to 0.87 c/° reading from the corridor midline).
In figure 13, the hovercraft can be seen to follow either the right (red or black trajectory) or the left wall (blue trajectory), depending on the sign of the error signal ε_{side}, (Eq. 1). The robot can be seen to generate a steady state clearance of 0.24m from either wall (figure 13a), while reaching a steady state forward speed of $V_x = 1m/s$ (figure 13b). Thus, the hovercraft adopts a wall-following behavior in much the same way as honeybees do in a similar situation (Serres et al., 2008a).

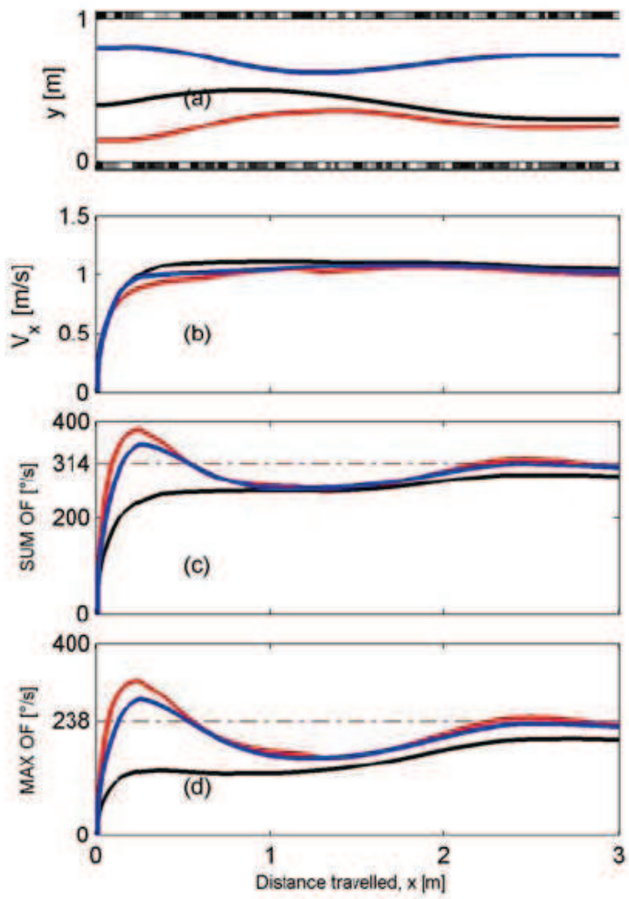


Figure 13. Hovercraft automatic wall-following behavior: (a) Simulated trajectories starting from three different initial positions y_0 (red: $y_0 = 0.15m$; black: $y_0 = 0.40m$; blue: $y_0 = 0.80m$). (b) Forward speed profiles. In the steady state, the forward speed can be seen to have reached $V_x = 1m/s$ in the three cases. (c) Sum of the two lateral optic flows ($\omega_R + \omega_L$). (d) Larger value of the two lateral optic flows, right and left

6.2.2 Automatic response in a tapered corridor

For the dual OF regulator, a tapered corridor acts like a *non-constant OF disturbance*. The forward control system adjusts the forward speed proportionally to the local corridor width (the width varies from 1.24m to 0.50m). The lateral control system controls the distance to the right wall in proportion to the forward speed at all times. This simulation experiment shows that the dual OF regulator is able to cope with the major disturbance caused by a tapered corridor, by making the robot decelerate or accelerate appropriately, figure 14.
The hardware co-simulation results presented here closely match the *Simulink* results presented by Serres et al. (2008a).

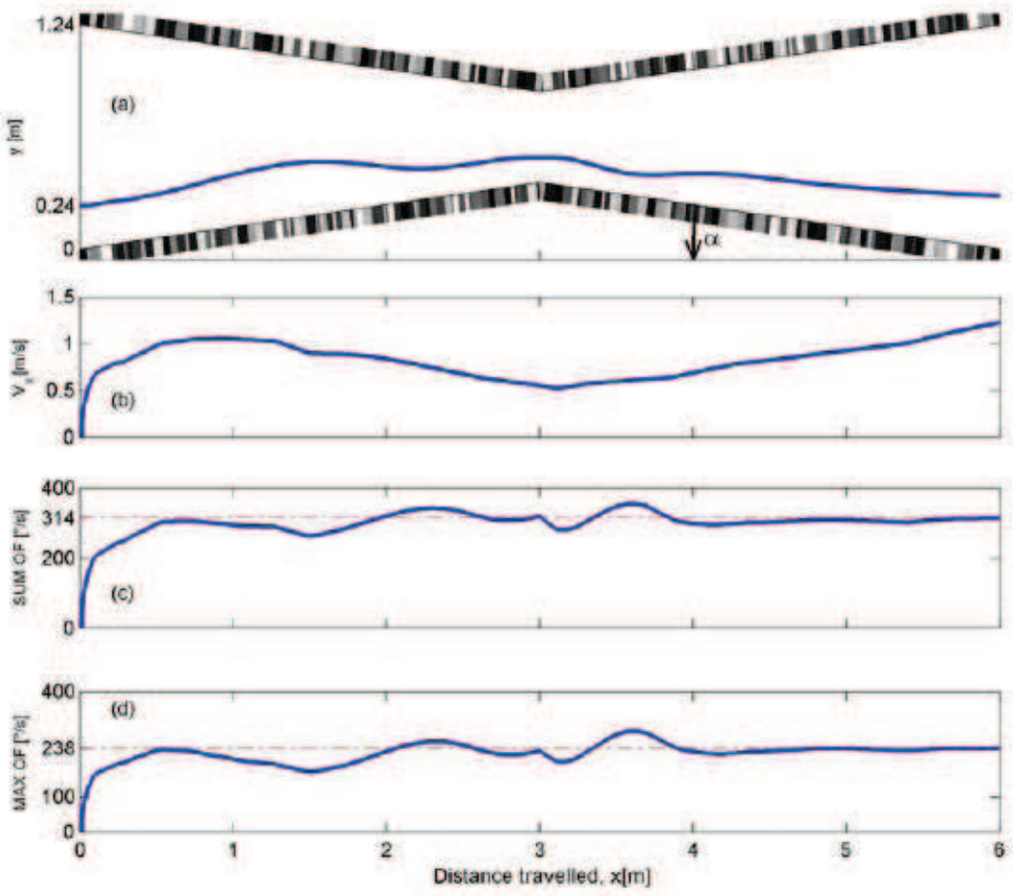


Figure 14. Automatic response in a *tapered* corridor: (a) Simulated trajectory of the hovercraft moving to the right in a tapered corridor starting at the initial position y_0 = 0.24m. This trajectory shows that the hovercraft automatically slows down when the local corridor width decreases and accelerates again when it widens again after the constriction.
(b) Corresponding forward speed profile V_x. The forward speed turns out to be a quasi linear function of the distance travelled x, and hence of the local corridor width. (c) Sum of the two optic flow ($\omega_R + \omega_L$). The speed control system succeeds to keep the sum of the optic flows measured virtually constant. (d) Larger value ("max") between the two optic flows measured. The side control system succeeds to keep the larger value of the two optic flows measured virtually constant

6.3 Hardware integration

In order to make the interface between the peripheral components and the FPGA-integrated system, several pilots have being developed and tested: driver ADC108S102, driver ADC101S101, driver DAC1101S101, PWM generator. These drivers link the components or the corresponding boards to the integrated *visuo-motor control loops* (generated by *System Generator*). They are integrated in the Virtex-4 FPGA and designed in VHDL because this language is more adapted to the realization of both the PWM generator and the communications protocols, such as the SPI protocol (*Standard Peripherical Interface*) that is used by ADC and DAC. For example, it is easier to manage the timing aspects of these low level hardware functions by a VHDL Finite State Machine synchronized by the FPGA clock.

The complete description can be performed in two different ways: (i) In *Simulink*, the *visuo-motor control system* is associated to import some drivers' descriptions (using Xilinx black box in the *System Generator* library); (ii) in Xilinx ISE environment, the VHDL generated description of the *visuo-motor control system* is connected to drivers descriptions (using ISE schematic description). The first solution (i) is difficult to apply because it requires the addition of peripheral components or functions high level models. This solution is not likely to improve the digital integration of control loops (validated in co-simulation) or drivers (validated in low level simulations and/or tested with peripheral components). The second solution does not allow any simulations of the closed loop system, but facilitates the integration stages (synthesis, mapping, placement and routing) and the generation of FPGA configuration binary file.

Table 2 shows the working characteristics of the Virtex-4 FPGA obtained after the integration of the LORA III autopilot and drivers.

DSP48	15 out of 32	42%
Slices	2569 out of 5472	47%
Block RAM 18kb	9 out of 36	25%

Table 2. Working characteristics of the XC4FX12 Virtex-4 FPGA

The DSP48 is a DSP-oriented component. The DSP48 is basically a multiplier followed by an adder with several optional registers on the ports and between the multiplier and adder. The multiplier takes two 18-bit signed signals and multiplies them, giving a 36-bit result. This is then sign extended to 48 bits and can be fed into the adder or routed directly to the outputs of the DSP48. The adder, which can be configured either as an adder or a subtractor, can accept the sign-extended output of the multiplier and 48-bit input to the DSP48. In addition, the adder can also accept itself as an input, to form an accumulator.

A slice is a basic element of Configurable Logic Block (CLB). A CLB element contains four interconnected slices. These slices are grouped in pairs. The elements common to both slice pairs are two logic-function generators (or look-up tables), two storage elements, wide-function multiplexers, carry logic, and arithmetic gates. The slices are used to provide logic, arithmetic, and ROM functions. Virtex-4 device features a large number of 18 Kb block RAM memories. True Dual-Port RAM offers fast blocks of memory in the

device. Block RAMs are placed in columns, and the 18 Kb blocks are cascadable to enable a deeper and wider memory implementation, with a minimal timing penalty.

7. Conclusion and future work

We have developed a digital integration of an autopilot called LORA III (Serres et al. 2008a) onto a Virtex-4 FPGA. The autopilot consists of two interdependent visuo-motor control loops that are meant to control the visual guidance of a miniature hovercraft in a corridor without any measurements of speed and distance from the walls.

A top-down methodology was used to design and simulate the overall visuo-motor control system. The latter was studied using both a high-level graphical environment: *Simulink from Mathworks,* and *System Generator for DSP* toolbox, from Xilinx. The remarkable analysis capability is due to the fact that *System Generator* allows the designed *sensory-motor control loop* to be implemented from within the *Simulink* environment. The design flow has simplified the integration problems in using several levels of abstraction that were validated at each stage of development. According to this methodology, digital specifications and architectures of each control loop were optimized for the LORA III autopilot. Moreover, final tests were performed by exploiting the hardware co-simulation. We were therefore able to test final descriptions in FPGA from *Matlab/Simulink* environment with a JTAG connection. In this way, integrated architectures were validated by considering the hovercraft "flying" in straight or tapered corridors.

Integrating the *visuo-motor control loops* also required designing a specific electronic board based on a Virtex-4 FPGA (Figure 12). Linking the control system to the external components (ADC, DAC, motors control) also required designing several drivers that were embedded into the same 0.5 gram FPGA.

Future work will consist in installing the FPGA based sensory-motor control board into the miniature hovercraft for which it was built. Tests similar to those made in co-simulation will then be carried out. Additional improvements are also planned to increase the robustness of LORA III control system and make the robot negotiate more challenging corridors. The passive OF sensors and the simple processing system described here are particularly suitable for use with Micro-Air Vehicles (MAVs), in which highly stringent constraints are imposed in terms of the permissible avionic payload and onboard energy resources. FPGA implementation has recently been envisioned not only for the visuo-motor control of micro-air vehicles but also for the automatic visual guidance and retrorocket control of future planetary landers.

8. References

Ahmad, Z. & Taib, M.N. (2003). A study on the dc motor speed control by using back-emf voltage, *Proceeding of IEEE Asian conference on Sensors (AsiaSense)*, pp. 359-364

Aubépart, F. & Franceschini, N. (2005). Optic flow sensors for robots: Elementary Motion Detectors based on FPGA, *Proceedings of IEEE International Workshop on Signal Processing Systems*, pp.182-187, Athens, Greece, 2-4 November, 2005

Aubépart, F. & Franceschini, N. (2007). Bio-inspired optic flow sensors based on FPGA: Application to Micro-Air-Vehicles. *Journal of Microprocessors and Microsystems*, Vol. 31, No.6, (2007) (408-419) , ISSN°0141-9331

Baird, E.; Srinivasan, M. V.; Zhang, S. & Cowling, A. (2005). Visual control of flight speed in honeybees. *The Journal of experimental Biology,* Vol.208, No.20, (3895-3905), ISSN°0022-0949

Blanes, C. (1986). Appareil visuel élémentaire pour la navigation à vue d'un robot mobile autonome, DEA thesis (in french), Aix-Marseille University

Browy, C.; Gullikson, G & Indovina, M. (1997). A Top-Down approach to IC design, www.indovina.us/~mai/a_top_down_approach_to_ic_design.pdf

Chambers, P. (1997). The ten commandments of excellent design: VHDL code examples, *Electronic design*, Vol. 45, No.8, (123-126)

Chalimbaud, P. & Berry, P. (2007). Embedded active vision system based on an FPGA architecture. *EURASIP Journal on embedded systems*, Vol.2007, No.1 (January 2007) (12p.), ISSN°1687-3955

Collett, T.S. (1980). Some operating rules for the optomotor system of a hoverfly during voluntary flight. *Journal of Comparative Physiology A*, Vol.138, No.3, (September 1980) (271-282), ISSN°0340-7594.

David, C. (1978). The relationship between body angle and flight speed in free-flying *Drosophila*. *Physiological Entomology*, Vol.3, No.3, (191-195), ISSN°0307-6962

Franceschini, N. (1985). Early processing of color and motion in a mosaic visual system, *Neurosciences. Res. Suppl*. 2, (17-49)

Franceschini, N.; Blanes, C & Oufar, L. (1986). Passive non-contact optical velocity sensor, *Dossier technique N° 51549* (in French), Agence Nationale pour la Valorisation de la Recherche/Direction de la Valorisation, Paris

Franceschini, N.; Riehle, A. & Le Nestour, A. (1989). Directionally Selective Motion Detection by insect neurons, In: *Facets of Vision*, Stavenga D.G. & Hardie R.C. (Eds.), (360-390), Springer, ISBN°0387503064, Berlin

Franceschini, N.; Pichon, J-M. & Blanes, C. (1992). From insect vision to robot vision, *Philosophical transactions of the Royal Society of London. Series B, Biological sciences* Vol. 337, No.1281, (283-294), ISSN°0080-4622

Franceschini, N.; Pichon, J-M. & Blanes, C. (1997). Bionics of visuomotor control, *In Evolutionary robotics: from intelligent robots to artificial life, AAAI books*, : T. Gomi (Ed.), (49-67), Ottawa

Franceschini, N. (1998). Combined optical, neuroanatomical, electrophysiological and behavioural studies on signal processing in the fly compound eye, In: *Biocybernetics of vision: Integrative Mechanisms and Cognitive Processes,* C. Taddei-Ferretti, (Ed.), (341-361), World Scientific, London

Golson, S. (1994). State Machine design techniques for Verilog and VHDL, *Synopsys Journal of High-Level Design*, (September 1994), (1-48)

Hardie, R.C. (1985). Functional organisation of fly retina, In: *Progress in Sensory Physiology*, D. Ottoson (Ed.), Vol.5 (1-79), Springer, Berlin

Iida, F. (2001). Goal-Directed Navigation of an Autonomous Flying Robot Using Biologically Inspired Cheap Vision, Proceedings of the 32nd International Symposium on Robotics (ISR), pp.1404-1409, 19-21 April 2001

Kennedy, J.S. (1939). Visual responses of flying mosquitoes. *Proceedings of Zoological Society of London,* Vol.109, 221-242.

Kennedy, J.S. (1951). The migration of the desert locust (Schistocerca gregaria Forsk.) I. The behaviour of swarms. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*,. Vol.235, No.625 (May 31, 1951), (163-290), ISSN°02610523.

Kerhuel, L.; Viollet, S. & Franceschini, N. (2007). A sighted aerial robot with fast gaze and heading stabilization, *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp.2634-2641, ISBN°978-1-4244-0912-9, San Diego, USA, October 29-November 2 2007

Kirchner, W.H. & Srinivasan, M.V. (1989). Freely flying honeybees use image motion to estimate object distance. *Naturwissenschaften*, Vol.76, No.6, (281-282), ISSN 0028-1042.

Koenderink, J.J. (1986) Optic flow, *Vision Research*, Vol.26, No.1, (161-179), ISSN°0042-6989

Kramer, J.; Sarpeshkar, R. & Kock, C. (1997). Pulse-Based Analog VLSI Velocity Sensors, *IEEE Transactions on Circuits and Systems* II, Vol.44, No.2 (February 1997) (86-101), ISSN° 1057-7130

Liu, S.C. & Usseglio-Viretta, A. (2001). Fly-like visuomotor responses of a robot using a VLSI motion-sensitive chips, *Biological Cybernetics*, Vol.85, No.6 (december 2001) (449-457), ISSN°0340-1200

Mura, F. & Franceschini, N. (1996). Obstacle avoidance in a terrestrial mobile robot provided with a scanning retina, In: *Intelligent Vehicles Vol.II*, M. Aoki and I. Masaki (Eds.), (47-52), MIT Press, Cambridge

Murthy, S. N.; Alvis, W.; Shirodkar, R.; Valavanis, K. & Moreno, W. (2008). Methodology for implementation of Unmanned vehicle control on FPGA using System generator, *Proceedings of the 7th international Caribbean conference on devices, circuits and systems (ICCDCS)*, pp.1-6, ISBN°978-1-4244-1956-2, Mexico, April 28-30, 2008

Netter, T. & Franceschini, N. (1999). Neuromorphic optical flow sensing for nap-of-the-earth flight, *Proceeding of SPIE Conference on Mobile Robots XIV*, Vol.3838, pp.208-216, Boston, USA, 20 September 1999

Netter, T. & Franceschini, N. (2002). A Robotic aircraft that follows terrain using a neuromorphic eye, *Proceeding IEEE International Conference on Robots and Systems (IROS)*, pp.129-134, ISBN 0-7803-7398-7, Lausanne, Swiss, 30 September - 4 October 2002.

Ownby, M. & Mahmoud W. H. (2003). A design methodology for implementing DSP with Xilinx System Generator for Matlab, *Proceedings of the 35th Southeastern Symposium on System Theory*, pp. 404-408, 16-18 March 2003

Pichon, J-M.; Blanes, C. & Franceschini, N. (1989). Visual guidance of a mobile robot equipped with a network of self-motion sensors, *Proceeding of SPIE Mobile RobotsIV.*, Vol.1195, (44-53), ISBN°9780819402349

Ruffier, F. & Franceschini, N. (2003). OCTAVE, a bioinspired visuo-motor control system for the guidance of Micro-Air-Vehicles, *Proceeding of SPIE Conference on Bioengineered and Bioinspired Systems*, Vol.5119, pp.1-12, Bellingham, USA

Ruffier, F.; Viollet, S. & Franceschini, N. (2004). Visual control of two aerial micro-robots by insect-based autopilots, *Advanced Robotics*, Vol.18, No.8, (771-786)

Ruffier, F. & Franceschini, N. (2005) Optic flow regulation: the key to aircraft automatic guidance, *journal of Robotics and Autonomous Systems*, Vol.50, No.4, (177-194), ISSN° 0921-8890

Ruffier, F., Serres J., Masson, G.P. and Franceschini N. (2007). A bee in the corridor: regulating the optic flow on one side. P*roceedings of the 7th Meeting of the German Neuroscience Society - 31st Göttingen Neurobiology Conference*, Göttingen, Germany, Abstract T14-7B.

Serres, J., Ruffier, F. & Franceschini, N. (2006). Two optic flow regulators for speed control and obstacle avoidance, *Proceedings of the first IEEE International Conference on Biomedical and Biomechatronics (Biorob)*, pp.750-757, Pisa, Italy.

Serres, J., Ruffier, F., Masson, G.P. & Franceschini N. (2007). A bee in the corridor: centring or wall-following? In *proceedings of the 7th Meeting of the German Neuroscience Society - 31st Göttingen Neurobiology Conference*, Göttingen, Germany, Abstract T14-8B.

Serres, J., Dray, D., Ruffier, & Franceschini N. (2008a). A vision-based autopilot for a miniature air vehicule: joint speed control and lateral obstacle avoidance. *Autonomous Robots*, Vol.25, No.1-2 (August 2008) (103-122). ISSN° 0929-5593

Serres, J., Masson, G.M., Ruffier, F. & Franceschini, N. (2008b). A bee in the corridor: centering and wall-following. Naturwissenchaften. (in press)

Srinivasan, M.V., Zhang, S.W., Lehrer, M. and Collett T.S. (1996). Honeybee navigation *en route* to the goal: visual flight control and odometry. *Journal of Experimental Biology*, Vol.199, (237-244)., ISSN°0022-0949.

Srinivasan, M.V., Zhang, S.W., Chahl, J.S., Barth, E., Venkatesh, S. (2000) How honeybees make grazing landings on flat surface. Biological Cybernetics, Vol.83, No.3, (171-183), ISSN°0340-1200

Srygley, R.B., and Oliveira, E.G. (2001). Orientation mechanisms and migration strategies within the flight boundary layer. In Insect Movements: Mechanisms and Consequences, T.P. Woiwod, D.R. Reynolds, and C.D. Thomas, (Eds). (183–206), Wallingford, Oxon, UK: CABI Publishing, CAB International), ISBN 0851994563.

Tammero, L.F. and Dickinson, M.H. (2002). The influence of visual landscape on the free flight behavior of the fruit fly *drosophila melanogaster*. *Journal of Experimental Biology*, Vol.205, No3 (327-343), ISSN°0022-0949

Turney, R. D.; Dick, C.; Parlour, D. P. & Hwang, J. (1999). Modeling and implementation of DSP FPGA Solutions, *Proceedings of International Conference on Signal Processing Applications and Technology (ICSPAT)*, November 1-4, 1999, Orlando, USA (http://www.xilinx.com/products/logicore/dsp/matlab_final.pdf)

Van Nieuwstadt, M. & Morris, J.C. (1995). Control of rotor speed for a model helicopter: a design cycle, *Proceedings of American Control Conference*, Vol.1, pp. 668-672

Viollet, S. & Franceschini, N. (2001). Aerial Minirobot that stabilizes and tracks with a bio-inspired visual scanning sensor, In: *Biorobotics: Methods and applications*, B. Webb and T. Consi (Eds.), (67-83), MIT Press, ISBN°026273141X, Cambridge

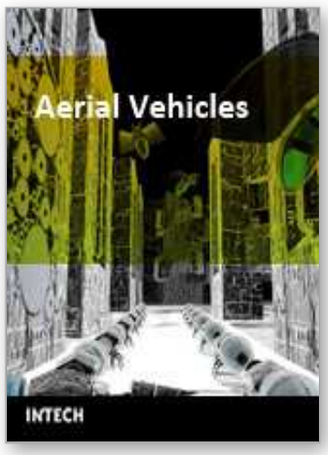
Viollet, S.; Kerhuel, L. & Francechini, N. (2008). A 1-gram dual sensorless spedd governor for Micro-Air-Vehicles, *Proceedings of 16th Mediterranean Conference on Control and Automation*, pp.1270-1275, ISBN°978-1-4244-2504-4, Ajaccio, France, June 25-27, 2008

Wagner, H. (1982). Flow-field variables trigger landing in flies. *Nature*, Vol.297, (147-148).

William C. (1965). *Insect migration*, Second edition ed. Collins editor, London.

Yamada, H.; Tominaga, T. & Ichikawa, M. (2003). An autonomous flying object navigated by real-time optical flow and visual target detection, *Proceedings of the IEEE International Conference on Field Programmable Technology*, pp.222-227, ISBN°0-7803-8320-6, Tokyo, 15-17 December 2003

Zufferey, J.C.; Beyeler, A. & Floreano, D. (2003). Vision-based Navigation from Wheels to Wings, *Proceedings of IEEE International Conference on intelligent Robots and Systems*, pp.2968-2973, ISBN°0-7803-7860-1, Las-Vegas, USA, 27-31 October 2003



Aerial Vehicles

Edited by Thanh Mung Lam

ISBN 978-953-7619-41-1
Hard cover, 320 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009

This book contains 35 chapters written by experts in developing techniques for making aerial vehicles more intelligent, more reliable, more flexible in use, and safer in operation.It will also serve as an inspiration for further improvement of the design and application of aeral vehicles. The advanced techniques and research described here may also be applicable to other high-tech areas such as robotics, avionics, vetronics, and space.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Fabrice Aubepart, Julien Serres, Antoine Dilly, Franck Ruffier and Nicolas Franceschini (2009). Field Programmable Gate Array (FPGA) for Bio-Inspired Visuo-Motor Control Systems Applied to Micro-Air Vehicles, Aerial Vehicles, Thanh Mung Lam (Ed.), ISBN: 978-953-7619-41-1, InTech, Available from: http://www.intechopen.com/books/aerial_vehicles/field_programmable_gate_array__fpga__for_bio-inspired_visuo-motor_control_systems_applied_to_micro-a

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen