Evolving Neural Networks for the Control of a Lenticular Blimp

Stéphane Doncieux and Jean-Arcady Meyer

AnimatLab - LIP6, France
http://animatlab.lip6.fr
{Stephane.Doncieux,Jean-Arcady.Meyer}@lip6.fr

Abstract. We used evolution to shape a neural controller for keeping a blimp at a given altitude, and as horizontal as possible, despite disturbing winds. The blimp has a lenticular shape whose aerodynamic properties make it quite different from a classical cigar-shaped airship. Evolution has exploited these features to generate a neural network that proved to be more efficient than a hand-designed PID-based controller that independently controlled the blimp's three degrees of freedom.

Keywords: neural networks, evolution, lenticular blimp

1 Introduction

Over about the past ten years, attempts at evolving neural controllers for robots have proliferated (see [4] for a review), and this approach is currently the most popular in evolutionary robotics ([6, 5]). However, it mostly involves crawling, rolling or walking robots, i.e., robots that move on the ground, and much more rarely swimming or flying robots, probably because such robots are still uncommon in academic laboratories. This situation may well evolve quickly, at least as far as aerial robots are concerned, if only because of the growing military and civilian needs for machines as small and as energetically economical as possible - a set of qualities that perfectly fits academic constraints. Moreover, the control of robots moving in a 3D-environment, whose complex dynamics are likely to be affected by wind or other disturbances, and whose sensory-motor equipment may well be limited by the size and energy constraints just alluded to, raises new and interesting challenges that will certainly trigger a number of future research efforts.

Be that as it may, previous attempts at evolving neural controllers for flying robots have been made by Doncieux [1, 2] and by Zufferey et al. [7]. The former work produced neural networks able to combat the effects of wind and to maintain either a constant flying speed and direction in a simulated blimp, or a constant position with respect to the ground in a simulated helicopter. The latter one produced neural controllers that moved a real blimp around a room and used visual information to detect collisions with walls.

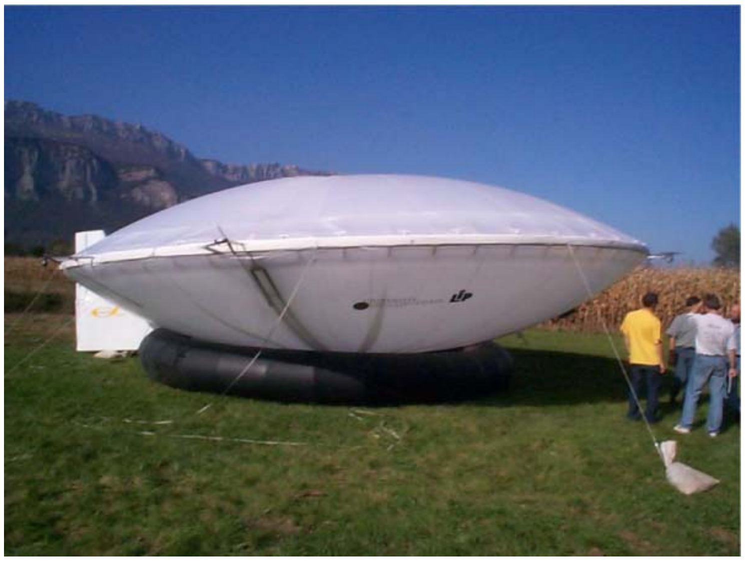

Fig. 1. A 10-meters wide lenticular blimp.

In the experiments reported here, we used an evolutionary algorithm to design a neural network that controls roll, pitch and altitude together in a simulated lenticular blimp. Although the ultimate goal of this research is to design an entirely automatic pilot, these experiments mostly contribute to an intermediate objective, i.e., to design a system that will help a human to pilot a real blimp 10-meters wide (figure 1). Because this engine has up to ten effectors, direct control of each motor would be too difficult for a human, and the evolved neural controller will help him in this task. The pilot will just need to set the desired altitude and to control the horizontal position of the aircraft, thus reducing the number of commands from ten to only three.

The article first describes the simulation model and the evolutionary approach that were used. Then, it describes the results that were obtained and provides details about the inner workings of an efficient controller. Directions for future work are also indicated.

2 Simulation model

A lenticular shape affords several advantages over the traditional cigar-shaped configuration that characterizes most blimps. In particular it renders the aircraft less prone to wind perturbations, thus allowing it, for instance, to be parked outside, directly tied to the ground, whereas cigar-shaped blimps, that need to be linked to the ground by some cable, cannot withstand high winds outside a hangar. However, the dynamic behavior of a lenticular blimp is much more complicated than that of a cigar-shaped blimp, thus providing a richer set of

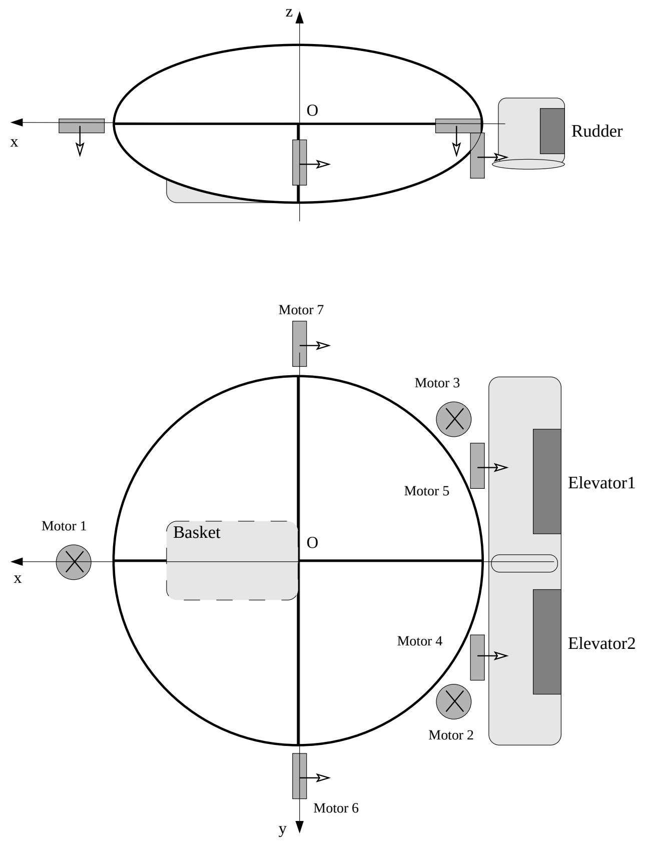

Fig. 2. Side and above views of the blimp's model. In the experiments reported here, only motors 1 to 3 were used.

interactions between the robot and its environment, on the one hand, but entailing greater efforts to devise realistic simulation models of such behavior, on the other.

The actual blimp that this research is centered around is equipped with two inclinometers - that are sensitive respectively to roll and pitch -, with an altimeter, a GPS, an anemometer, and a video camera. The blimp is also equipped with seven motors and three control surfaces (figure 2). Motors 1 to 3 are used to control pitch, roll and altitude. Motors 4 to 7 are used for propulsion, but motors 6 and 7 may also control the yaw of the aircraft if their thrusts are set antagonistically. Control surfaces serve the same purpose as motors, although they waste less energy, but they are not usable in all circumstances. In the work to be described here, we did not use them and concentrated on the control of altitude, pitch and roll using only the inclinometers and the altimeter, on the one hand, and motors 1 to 3, on the other.

The general equation that rules the dynamics of the model is:

$$M_t\boldsymbol{\Gamma} = M_t\boldsymbol{g} + \boldsymbol{P} + \boldsymbol{A}_{OXYZ} + \boldsymbol{F}_{OXYZ}$$

$\boldsymbol{A}$ is the aerodynamic force, $\boldsymbol{F}$ is the resultant of command forces, and $\boldsymbol{P}$ is Archimede's thrust. To simplify, we suppose that the aircraft is balanced, i.e., that the Archimede's thrust exactly compensates gravity. The resultant of aerodynamical forces acting on the blimp is computed according to equation:

$$\boldsymbol{A_{Gxyz}} = \begin{bmatrix} T_0 \cos i.\cos j + P_0.\sin i.\cos j \\ T_0.\cos i.\sin j + P_0.\sin i.\sin j \\ -T_0.\sin i + P_0.\cos i \end{bmatrix} \tag{1}$$

where T_0 is the drag and P_0 the lift, computed as follows:

$$T_0 = -\frac{1}{2}.\rho.S.V_{rel}^2.C_x \tag{2}$$

$$P_0 = \frac{1}{2}.\rho.S.V_{rel}^2.C_{z_i}.\sin(2.i) \tag{3}$$

i is the incidence and j the sideslip of the blimp, computed as follows:

$$i = \arcsin\left(-\frac{Vr_z}{V_{rel}}\right) \tag{4}$$

$$j = \arcsin\left(\frac{1}{\cos i}\frac{Vr_y}{V_{rel}}\right) \tag{5}$$

V_{rel} is the speed of the blimp relative to the surrounding air. Vr_x, Vr_y and Vr_z are its coordinates in the absolute reference. $\rho = 1.2255 kg/m^3$, S is the reference surface: $S = V^{\frac{2}{3}}$, V being the volume of the blimp.

Rotation speed and angles are computed as follows:

$$\psi^{'} = \frac{(q.\sin\varphi + r.\cos\varphi)}{\cos\theta} \tag{6}$$

$$\theta^{'} = q.\cos\varphi - r.\sin\varphi \tag{7}$$

$$\varphi^{'} = p + \psi^{'}.\sin\theta \tag{8}$$

$$p^{'} = \frac{1}{I_{x1}}.(L - (I_z - I_y).r.q) \tag{9}$$

$$q^{'} = \frac{1}{I_{y1}}.(M - (I_x - I_z).p.r) \tag{10}$$

$$r^{'} = \frac{1}{I_z}.(N - (I_y - I_x).p.r) \tag{11}$$

$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$ is the rotation vector around the center of gravity. θ is the pitch, φ the roll and ψ the yaw.

$I = \begin{bmatrix} Ix & 0 & 0 \\ 0 & Iy & 0 \\ 0 & 0 & Iz \end{bmatrix}$ is the matrix of inertia moments. It takes into account the respective contributions of the motors, the basket and the envelope.

The complete model is too complex to be described here in detail. Suffice it to say that the corresponding code is more than 3000-instructions long, that the mass of the air and the ground effect are taken into account, and that the wind is modelled via a Drydden spectrum. Furthermore, Mt is set to $115kg$ and V to $141m^3$, and the equations are integrated using a fourth order Runge-Kutta method.

Numerous simulations have already been performed that validated the qualitative behavior of the airship. Preliminary experiments with the real blimp, which aimed at assessing the correspondence between its behavior and that of the simulated model, have also been carried out and yield encouraging results.

3 Evolutionary procedure

The procedure that serves to evolve neural networks able to control the blimp calls upon an indirect encoding scheme that favors the discovery of symmetries and the reuse of useful modules. Only a simplified version of this general scheme, which will be described elsewhere, has been used here.

3.1 The chromosome

A chromosome encodes a list of modules, a list of links, and a list of template-weight values. In this simplified version, a module is composed of just a single connection between an input neuron and an output neuron, and links between modules fuse the output neuron of a module with the input neuron of another module. Other links serve to connect the input neuron of a module with a sensory neuron whose activity level equals the error currently detected by the blimp's corresponding sensor. Likewise, other links serve to connect the output neuron of a module with a motor neuron whose activity level, which varies between +1 and -1, modulates the maximum thrust that the blimp's corresponding motor exerts. By convention, these two categories of links are given connexion weights of +1. Figure 3 illustrates how a chromosome is decoded into a neural network that may exhibit symmetries and modular redundancies.

During the course of evolution, several mutation operators make it possible to create or suppress some template-weights, links or modules. A single crossover operator is used to exchange modules between chromosomes.

List of weight values

1: w=−3.1	4: w=−6.0
2: w=4.2	5: w=10.0
3: w=1.6	

List of modules

m1	m2	m3	m4	m5	m6	m7
3	2	1	1	5	2	2

List of links between modules, inputs and outputs

m1−>m3	m1−>m2
m2−>o0	m6−>o2
m3−>o1	m7−>o1
i1−>m4	m4−>m6
m4−>m7	i0−>m1

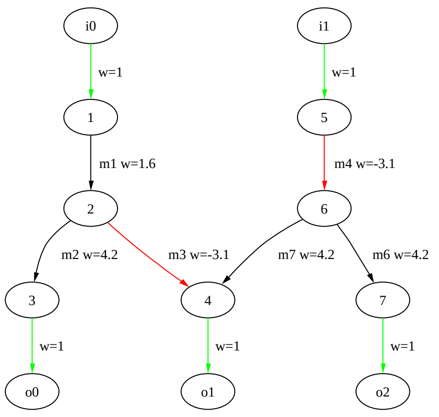

Fig. 3. The chromosome on the left of the figure codes for the neural network on the right, in which i_n are sensory neurons, and o_m are motor neurons. The chromosome includes three lists. The list of modules and the list of template-weights specify the synaptic weight associated with the inner connection of each module. The list of links specifies how a given module is connected to a sensory neuron, a motor neuron or to another module. Note that some modules or some weights may be defined in the corresponding lists, but without being actually used in the final network.

3.2 Experimental set-up

The task to be performed was both to maintain the blimp at a target altitude that changed over time, and to keep it as horizontal as possible, despite wind disturbance. Therefore, the fitness function we used takes into account the error on the three degrees of freedom we wanted to control, together with the energy consumption, in order to encourage low cost solutions. Its expression is:

$$f(t) = \frac{\sum_T (1 - \frac{|\delta\theta(t)|}{\delta\theta_max}) + \sum_T (1 - \frac{|\delta\varphi(t)|}{\delta\varphi_max}) + \sum_T (1 - \frac{|\delta z(t)|}{\delta z_max}) + \sum_T (1 - \frac{E(t)}{E_{max}})}{4 \times T}$$

in which $\delta\theta(t)$ and $\delta\varphi(t)$ are the pitch and roll errors with respect to 0, the desired values for these variables, and $\delta z(t)$ is the difference between the real altitude and the current target value. $E(t)$ is the total energy consumed during time step t. E_{max} is the maximum energy motors can consume during one time-step.

The neural networks that were generated could include three sensory neurons whose activity levels were respectively equal to $\delta\theta(t)$, $\delta\varphi(t)$ and $\delta z(t)$, and three motor neurons whose activity levels determined the command forces. In other words, the neural controllers could use the information provided by the blimp's inclinometers and altimeter to set the thrust that each of the three vertical motors should exert.

Using a classical GA algorithm, we ran experiments lasting 1000 generations with a population of 100 individuals to generate efficient controllers. The maximum number of modules that could be included in a given controller was set to 20. The mutation and crossover rates were empirically chosen and did not pretend to optimality. Template-weights were coded on 8 bits and varied between -10 and 10. Each neuron's transfert function was a simple $tanh$ function. Fitness evaluations were performed along four different runs, each lasting 4 min 10 s of simulated time, a period near the middle of which the wind conditions were changed, the corresponding speed and orientation being randomly chosen between respectively $0m/s$ and $6m/s$ and $0°$ and $360°$. However, because of the rudder, the blimp passively lined itself up with the wind direction in approximately 2 seconds. Likewise, at some instant during each evaluation run, the desired altitude was also randomly changed to some value between $100m$ and $200m$, while the roll and pitch errors were abruptly perturbed around their target 0 values. In other words, the fitness of each controller was assessed through 1000 sec of simulated time, a period during which both the wind conditions and the control objectives were changed four times.

These experiments were performed using the SFERES framework [3] which makes it easy to change every major option in the simulation set-up, be it the evolutionary algorithm, the genotype-phenotype coding, the simulation model or the fitness evaluation procedure.

4 Results

4.1 Observed behavior

Figure 4 shows the behavior generated by the best controller evolved in this manner. It thus appears that, following each imposed disturbance, the altitude (z) and roll (φ) are effectively kept at desired values, whereas the pitch (θ) temporarily deviates from the desired null value, although it eventually does return to it. This behavior exploits the dynamic properties of the blimp, whose lenticular shape makes it behave like a wing in the wind. Thus, when the blimp has to go up or down, the pitch angle must be negative in the former case and positive in the latter[1]. This makes it that, in order to improve fitness, the horizontal trim of the blimp must not be maintained during the whole experiment. Similar behavior has been observed with every controller getting a high fitness rating.

4.2 Comparison with a hand-designed controller

To evaluate the efficiency of this strategy, we hand-designed a reference controller whose behavior was compared to that of the evolved network. This reference controller called upon three PID modules that separately managed each degree of freedom. The input of each such PID module was the error of the degree of freedom it was supposed to keep, while its output was sent to an interface that set the three motor thrusts. Thus, the altitude error generated identical thrusts

[1] it is traditional in aeronautics to count as positive the angle of an aircraft steering towards the ground.

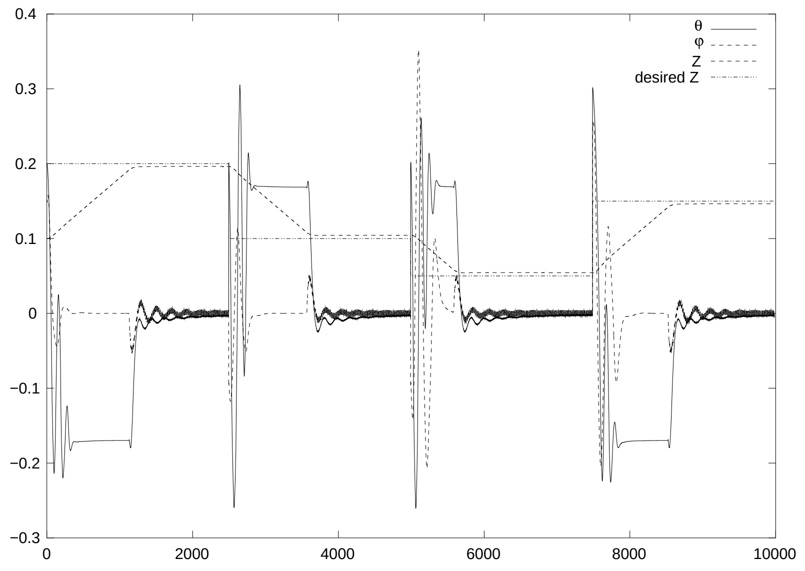

Fig. 4. Typical behavior generated by the best selected neural controller. θ (pitch angle) and φ (roll angle) must be kept at 0 (corresponding to a horizontal trim), while the altitude must be adjusted to a target value. Wind conditions, altitude target, and pitch and roll values are randomly changed every 2500 time-steps. It should be noted that, when the blimp has to go up or down in order to reach the new target value, the pitch is not kept at zero, but around a non-null value (approximately 0.17 radians, i.e. 10 degrees) whose sign depends upon the vertical direction the blimp has to follow. A time-step corresponds to 25 ms, angles are in radians and altitude in meters (altitudes are divided by 1000).

at the level of each motor, the roll error generated thrusts of opposite signs for motors 2 and 3, and the pitch error generated thrusts of opposite signs for motor 1, on the one hand, and for motors 2 and 3, on the other hand. The three inner parameters of each PID module were also optimized with an evolutionary algorithm.

A thorough comparison of the behaviors respectively generated by a hand-designed controller and an evolved neural network reveals that the former, which does not call upon any dynamic specificity of the lenticular shape, always maintains the horizontal trim of the blimp, a behavior fundamentally different from that of the evolved neural controller. However, the neural controller is about three times faster than the hand-designed one as far as the control of altitude is concerned (figure 5) because it is able to exploit the dynamic couplings between the blimp's degree of freedom, as explained in the next subsection.

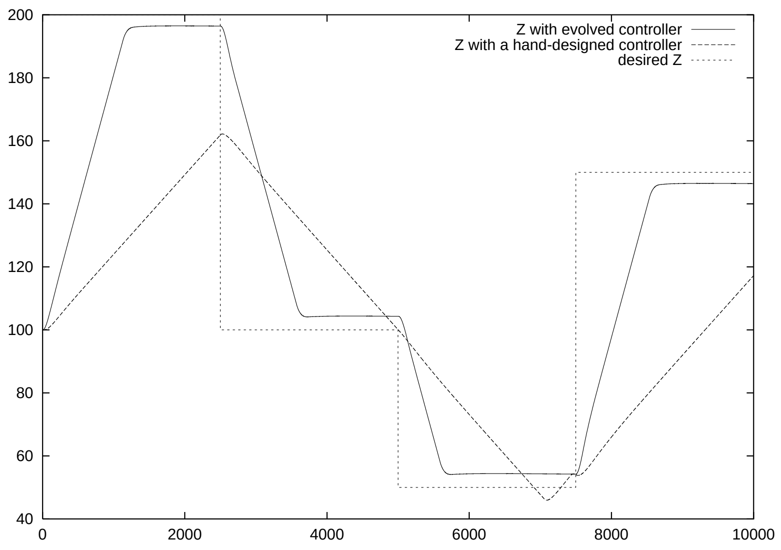

Fig. 5. Comparison of the behaviors generated by an evolved neural network and by a hand-designed system with respect to the control of altitude. The evolved controller is three times faster than the hand-designed one. A time step corresponds to 25 ms, altitudes are given in meters.

4.3 Experimental study of the neurocontroller

Figure 6 shows the evolved neurocontroller that generated the experimental results presented above. It consists of three sensory neurons, three motor neurons and eight hidden neurons, encapsulated into nine modules. Hidden neurons 2 and 3 exhibit a recurrent connection.

As a consequence of this structure, altitude control calls upon the three motors, as might have been expected. Pitch and roll control exploits only one motor, a solution that would probably not be chosen by a human designer, but which results in nearly the same effect on the blimp control.

In particular, it appears that, when there is no error in altitude, neuron 2 oscillates at a constant frequency, with a constant amplitude. At the level of neuron 1, these oscillations are modulated in amplitude by the activity of neuron 7 (figure 7). On average, these oscillations force the blimp to revert to a null pitch.

When there is an error in altitude, the activity value of neuron 2 always remains saturated, either at +1 - when the blimp is above the target altitude - or at -1 - when the blimp is below the desired altitude. This constant value acts as a bias on neuron 1, thus shifting the equilibirum point of pitch control from zero to a non null value whose sign depends on the sign of the error on altitude. Motor 1 accordingly inclines the blimp in order to generate the force that will

List of weights

0: w=2.5	5: w=7.08
1: w=−1.41	6: w=1.49
2: w=−8.89	7: w=−3.46
3: w=−1.65	8: w=−8.74
4: w=−8.11	9: w=−1.25

List of modules

m0	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10
9	5	8	4	8	2	1	5	5	2	9

List of links

m6−>o3	m8−>m6	m1−>m2
m4−>o3	i1−>m8	m2−>m9
i0−>m7	i2−>m9	i2−>m3
m7−>o1	m9−>o2	m3−>m8
m1−>m4	m1−>m10	
i2−>m1	m10−>o1	

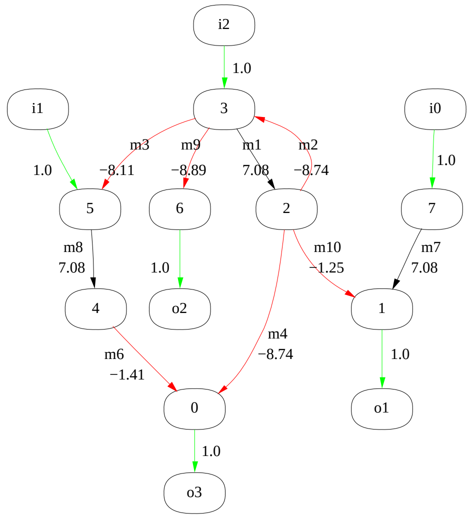

Fig. 6. Structure of the best neural network obtained (right) together with the corresponding chromosome (left). i_0,i_1 and i_2 are sensory neurons that measure pitch, roll and altitude errors. o_1, o_2, and o_3 are motor neurons that are used to set the blimp's horizontal trim and altitude.

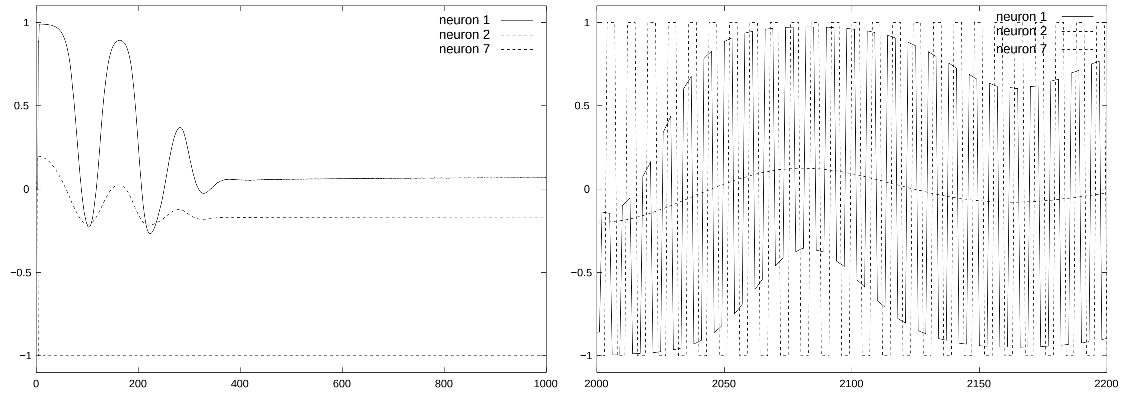

Fig. 7. Outputs of neurons 1, 2 and 7. Left: when the current altitude does not match the target value, the activity level of neuron 2 is saturated (here, it equals -1), thus acting as a bias on neuron 1. The activity level of this neuron, in turn, determines how much, and in which direction, the blimp will be inclined to help getting back to the target altitude. Right: when the altitude is correct, neuron 2 oscillates between its saturation values. Its activity, as well as that of neuron 1, is modulated by the output variations of neuron 7 to keep the blimp horizontal.

reduce the altitude error, as previously explained (figure 7). Similar dynamic couplings explain how roll error is controlled.

Concerning the oscillations, it can be shown that, when a pitch or roll error is detected, because of the influence of the recurrent connection between neurons 2 and 3, a periodic activity is generated at the level of neuron 0, and that the corresponding period (200 ms) is precisely the one that best exploits the time constants of the blimp's dynamic behavior. In other words, evolution discovered a solution that capitalizes on the complex sensory-motor coupling that a lenticular shape affords, and it ultimately converged towards a periodic signal that minimizes pitch and roll oscillations at the same time.

5 Discussion and directions for future work

The above controller basically calls upon a 'bang-bang' strategy that has probably been favored by the large range of values that could be assigned to connection weights (+10,-10), thus easily saturating the activation levels of the neurons. Although it is efficient in simulation, it should not be implemented on the real blimp because it would probably be too demanding at the motor level and liable to cause damage. New evolutionary runs will accordingly be carried out with a limited weight range.

Likewise, additional experiments are required to assess the respective roles of the various implementation choices that have been made here on a purely empirical basis, notably as far as mutation operators are concerned. This way, an optimal evolutionary set-up could be used to tackle more complex problems than those this article was focused upon. In particular, future experiments will involve the camera, and all available motors and control surfaces, at the same time, in order to minimize the energy spent in keeping the blimp at a given altitude, as horizontal as possible, and directly above a given objective. This will entail using a higher number of sensors and actuators than currently done in evolutionary robotics, in order to control up to five degree of freedom, and will in particular help in assessing the scalability of our approach.

Naturally, our ultimate goal is to implement the resulting controllers on the real blimp and to demonstrate their effectiveness.

6 Conclusion

This work demonstrates that it is possible to automatically evolve neural controllers able to exploit the highly specific dynamic couplings between a lenticular blimp's degrees of freedom. In particular, these controllers are more efficient than humanly designed ones that do not exploit such couplings, while remaining simple enough to be easily implemented on an on-board computer.

Acknowledgement

This work was part of the ALPhA project that aims at demonstrating the potential of lenticular aerial platforms. Besides the AnimatLab, it brings together Pierre Balaskovic - who designed the blimp blueprints and leads the project -, Samuel Tardieu, Gérard Mouret and Bertrand Dupouy from the École Nationale Supérieure des Télécommunications (ENST) - who designed the embedded electronics and the interface software - and AirStar Society - where the aircraft was built.

References

1. S. Doncieux. Évolution d'architectures de contrôle pour robots volants. In Drogoul and Meyer, editors, *Intelligence Artificielle Située*, pages 109–127, Paris, october 1999. Hermes.
2. S. Doncieux. Évolution d'architectures de contrôle pour robots volants. Master's thesis, LIP6 - ENSEA, 1999.
3. S. Landau, S. Doncieux, A. Drogoul, and J.-A. Meyer. Sferes: un framework pour la conception de systèmes multi-agents adaptatifs. *Technique et Science Informatique*, 21(4), 2002.
4. J.-A. Meyer. Evolutionary approaches to neural control in mobile robots. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Diego, 1998.
5. J.A. Meyer, S. Doncieux, D. Filliat, and A. Guillot. Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots. In Duro, Santos, and Graña, editors, *Biologically Inspired Robot Behavior Engineering.* Springer Verlag, 2002.
6. S. Nolfi and D. Floreano. *Evolutionary Robotics: Biology, Intelligence and Technology of Self-organizing Machines.* The MIT Press, 2001.
7. J.C. Zufferey, D. Floreano, M. van Leeuwen, and T. Merenda. Evolving vision-based flying robots. In Bülthoff, Lee, Poggio, and Wallraven, editors, *Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision LNCS*, 2002.