# Flying over the Reality Gap: From Simulated to Real Indoor Airships*

Jean-Christophe Zufferey, Alexis Guanella†, Antoine Beyeler and Dario Floreano

August 11, 2006

Laboratory of Intelligent Systems (http://lis.epfl.ch)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
Jean-Christophe.Zufferey@epfl.ch

**Abstract**

Because of their ability to naturally float in the air, indoor airships (often called blimps) constitute an appealing platform for research in aerial robotics. However, when confronted to long lasting experiments such as those involving learning or evolutionary techniques, blimps present the disadvantage that they cannot be linked to external power sources and tend to have little mechanical resistance due to their low weight budget. One solution to this problem is to use a realistic flight simulator, which can also significantly reduce experimental duration by running faster than real time. This requires an efficient physical dynamic modelling and parameter identification procedure, which are complicated to develop and usually rely on costly facilities such as wind tunnels. In this paper, we present a simple and efficient physics-based dynamic modelling of indoor airships including a pragmatic methodology for parameter identification without the need for complex or costly test facilities. Our approach is tested with an existing blimp in a vision-based navigation task. Neuronal controllers are evolved in simulation to map visual input into motor commands in order to steer the flying robot forward as fast as possible while avoiding collisions. After evolution, the best individuals are successfully transferred to the physical blimp, which experimentally demonstrates the efficiency of the proposed approach.

## 1 Introduction

Because of their ability to naturally float in the air, indoor airships (blimps) are widely used as research platform in aerial robotics. Many laboratories rely on such platforms for experiments in visual servoing [da Silva Metelo and Garcia Campos, 2003, van der Zwaan et al., 2002, Zhang and Ostrowski, 1998], collective intelligence [Melhuish and Welsby, 2002], bioinspired robotics [Iida, 2003, Planta et al., 2002, Bermúdez i Badia et al., 2005], or evolutionary robotics [Zufferey et al., 2002, Zufferey, 2005], but none of them propose a concise and pragmatic way of modelling and simulating such flying robots. However, a realistic and efficient simulator is crucial for long-lasting experiments such as those involving learning or artificial evolution. This transfer constitutes a major issue in learning and evolutionary approaches, which is often referred to as the "reality gap" (for a review see Nolfi and Floreano, 2000, section 3.3).

This article is structured in two parts. The first part presents the dynamic modelling together with the procedure for identifying its parameters without relying on costly facilities like wind tunnels. Our

*The original publication is available at www.springerlink.com

†Jean-Christophe Zufferey and Alexis Guanella contributed equally to this work.

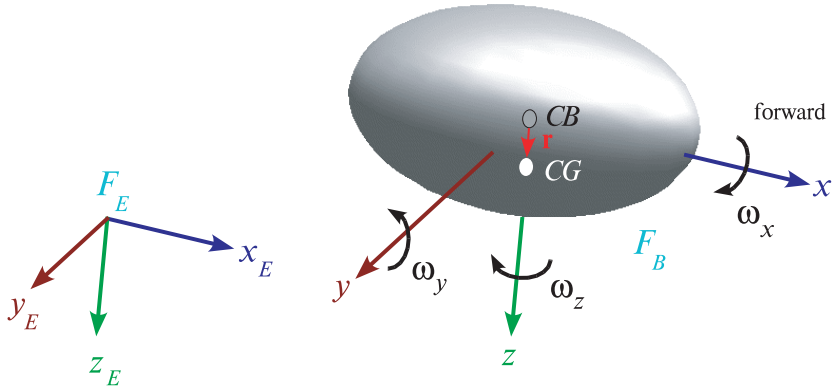


Figure 1: Inertial frame $\mathcal{F}_E$ and body-fixed frame $\mathcal{F}_B$. The origin of $\mathcal{F}_E$ is an arbitrary point on Earth surface, whose acceleration is neglected. Note that the $z_E$-axis is pointing downward. The origin of the body-fixed frame $\mathcal{F}_B$ is the center of buoyancy (CB), which corresponds to the center of the hull. Its orientation coincides with the orientation of the airship and its axes correspond to principal axes of the rigid body.

model is generic in the sense that it can virtually be used for any indoor airship with hull-shaped envelop flying at relatively low speed. The novelty in this first part does not lie in the mathematics of the model itself, which has been mainly adapted from fluid dynamics theory, but rather in the pragmatic methodology for swift identification of the different constitutive parameters of the model. In the second part, the dynamic model is validated by analysing the performance of airship controllers evolved in simulation and tested in reality. A well-studied task consisting of simple vision-based navigation (2D course stabilization and obstacle avoidance) in a randomly textured environment [Zufferey, 2005] has been chosen in order to demonstrate the usefulness of the proposed modelling approach in a practical situation. Actually, this methodology allowed us to dramatically speed-up the evolution process, which provided the basis for efficiently exploring different sets of control parameters that eventually led to significantly improved solutions with respect to the one described previously [Zufferey et al., 2002]. Furthermore, it is the first time, to the best of the author's knowledge, that such a simulation-reality transfer is successfully employed with a flying robot.

## 2 Airship Dynamic Model

Our dynamic model is specifically designed for hull-shaped indoor airships that can be approximated by an ellipsoid of revolution. The steering and propulsion is ensured by means of thrusters (e.g., DC motor with propeller), whose number, orientation and location can be freely chosen[1]. The typical maximum velocity of such aerial vehicles reaches 1m/s whereas their length is usually in the range of 1 to 3m. The hull and gondola distortions are assumed to be small and to have a negligible impact on the trajectory, allowing for considering the whole airship as a rigid body. The airship has two vertical planes of symmetry, in the intersection of which is located the center of gravity (CG) as well as the center of buoyancy (CB). These assumptions apply generally well to small airships with inflated envelope and lightweight gondola. Furthermore, they generate significant simplifications in the equation of motion and allow easier parameter identification.

In order to describe the motion of the airship in the 6 degrees of freedom (DOF) and the forces and moments acting on it, we define two reference frames: an earth inertial reference frame $\mathcal{F}_E$ and a body-fixed frame $\mathcal{F}_B$ (Fig. 1). The CG is located by the vector $\mathbf{r} = (0, 0, r_z)^T$ in $\mathcal{F}_B$.

[1]No control surfaces have been taken into account in our modelling because small indoor airships are rarely equipped with such steering means featuring poor efficiency at low velocity. However, it would not be an issue to model them similarly to aircraft wings and add their effects to our model.

Translational and rotational velocities $\boldsymbol{\nu} := (\boldsymbol{v}^T, \boldsymbol{\omega}^T)^T = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)^T$ are described in $\mathcal{F}_B$, whereas the position and orientation of the vehicle $\boldsymbol{\eta} := (x_E, y_E, z_E, \phi, \theta, \psi)^T$ are expressed in $\mathcal{F}_E$. The 3 last terms of $\boldsymbol{\eta}$ are the aeronautical Euler angles, namely roll $\phi$, pitch $\theta$, and yaw $\psi$.

The Newton-Euler equation of motion links the acceleration of the airship to the forces and moments acting on it. This non-linear equation is written using a vector representation for 6 DOF in $\mathcal{F}_B$:

$$\mathbf{M}\dot{\boldsymbol{\nu}} = \sum \mathbf{F}_{external} = \mathbf{F}_R + \mathbf{F}_P + \mathbf{F}_D + \mathbf{F}_C, \tag{1}$$

where the five main elements are listed below, following the order in which they will be presented:

$\mathbf{F}_R$: restoring forces (Section 2.1) containing gravity and buoyancy, which counteract each other and are responsible for maintaining the airship uright;

$\mathbf{F}_P$: propelling forces (Section 2.2), which are directly related to motor commands;

$\mathbf{F}_D$: damping forces (Section 2.3) due to air friction;

$\mathbf{M}$: inertia matrix (Section 2.4) containing rigid-body inertia and added mass terms;

$\mathbf{F}_C$: Coriolis and centripetal effects (Section 2.5), which are fictitious forces appearing in non-inertial frames such as $\mathcal{F}_B$.

The presentation order of those elements is motivated by both the identification process, which sometimes requires the value of previous components to be determined first (e.g. propelling thrusts are needed for the measurement of damping forces), and the fact that we use the inertia matrix as a basis for the derivation of the Coriolis matrix (Section 2.5).

## 2.1 Restoring Forces

Unlike airplanes, the aerostatic lift force (buoyancy) acting on an airship is independent of flight speed. The buoyant force is explained by the Archimedes' principle and is equal to the weight of the air displaced by the airship. Gravity and buoyancy together are called restoring forces because they are responsible for keeping the airship upright. Their amplitudes are expressed by:

$$F_g = mg \quad \text{and} \quad F_b = \rho V g, \text{ with } V = \frac{4}{3}\pi a b^2, \tag{2}$$

where $m$ is the mass of the airship, $g$ is the Earth gravitational acceleration, $\rho$ is the air density, and $V$ the volume of the ellipsoidal hull with semi-axes $a$ and $b$. The restoring forces are expressed in the body-fixed frame $\mathcal{F}_B$:

$$\mathbf{F}_R(\boldsymbol{\eta}) = \begin{pmatrix} -(F_g - F_b)\sin(\theta) \\ (F_g - F_b)\cos(\theta)\sin(\phi) \\ (F_g - F_b)\cos(\theta)\cos(\phi) \\ -r_z F_g \cos(\theta)\sin(\phi) \\ -r_z F_g \sin(\theta) \\ 0 \end{pmatrix}. \tag{3}$$

The distance $r_z$ (between CB and CG) can be identified by temporarily modifying the mass distribution along the $x$-axis and measuring the resulting static pitching angle. To do so, one can simply displace a subpart of the ballast, let say of a mass $m_1$, along the $x$-axis, at a distance $x_1$ from CB (Fig. 2). Then the resulting pitching angle $\theta_1$ allows for deriving $r_z$ using the following formula based on simple geometric considerations:

$$r_z = \frac{m_1}{m}\frac{x_1}{\tan(\theta_1)}. \tag{4}$$

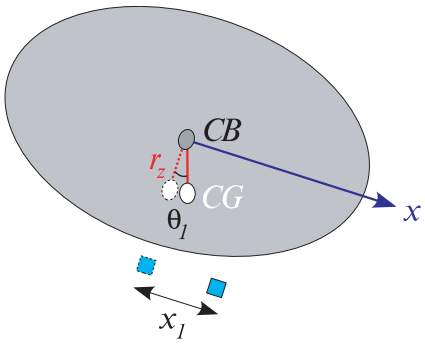


Figure 2: Procedure for the localization of the center of gravity. The mass distribution along the $x$-axis is temporarily modified, resulting in a pitch angle $\theta_1$

## 2.2 Propulsion

In our model, engines are assumed to be ideal thrusters, whose effect are directly proportional to the motor commands, and neither propeller fluxes nor motor torques are taken into account. Therefore, the propulsion matrix $\mathbf{F}_P$ depends only on motor commands together with the location and orientation of the engines. The identification procedure is thus limited to the determination of the thrust amplitude as a function of motor commands. The static thrust of each thruster can easily be measured for different motor commands, using a force sensor or a scale.

## 2.3 Damping

Aerodynamic damping is due to air friction, which depends on the velocity of the airship.[2] In general, there are two different regimes that can be distinguished: linear friction due to laminar boundary layers and quadratic friction due to turbulent boundary layers. Since it is difficult to know in advance in which regime the airship is operating, we model the damping as a second order Taylor series, accounting for both effects.

$$\mathbf{F}_D = \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu}, \text{ with } \mathbf{D}(\boldsymbol{\nu}) = -diag \begin{pmatrix} D_{v_x} + D_{v_x^2} |v_x| \\ D_{v_y} + D_{v_y^2} |v_y| \\ D_{v_z} + D_{v_z^2} |v_z| \\ D_{\omega_x} + D_{\omega_x^2} |\omega_x| \\ D_{\omega_y} + D_{\omega_y^2} |\omega_y| \\ D_{\omega_z} + D_{\omega_z^2} |\omega_z| \end{pmatrix}, \tag{5}$$

where $\mathbf{D}(\boldsymbol{\nu})$ is the damping matrix, $D_{v_x}$, $D_{v_y}$, $D_{v_z}$, $D_{\omega_x}$, $D_{\omega_y}$, $D_{\omega_z}$ are the linear damping coefficients, and $D_{v_x^2}$, $D_{v_y^2}$, $D_{v_z^2}$, $D_{\omega_x^2}$, $D_{\omega_y^2}$, $D_{\omega_z^2}$ the quadratic damping coefficients. This uncoupled model of damping is a rough approximation that works sufficiently well in case of low speed and highly symmetrical ellipsoid hull [Fossen, 1995].

The 12 damping coefficients can be identified by measuring stationary velocities reached by the airship subjected to different constant forces. For instance, a known thrust $F_{P_x}$ is applied in the forward direction by means of one or several engines with constant thrust (Fig. 3). When the blimp reaches a constant forward velocity, inertial and Coriolis effects are null and the following part of the equation of motion is used to identify the two damping coefficients related to the x-axis:

$$F_{P_x} - D_{v_x} v_x - D_{v_x^2} v_x^2 = 0, \text{ with } v_x > 0. \tag{6}$$

[2]This damping effect also account for apparent aerodynamic lift forces occurring when the airship is translating with a non-zero attitude.

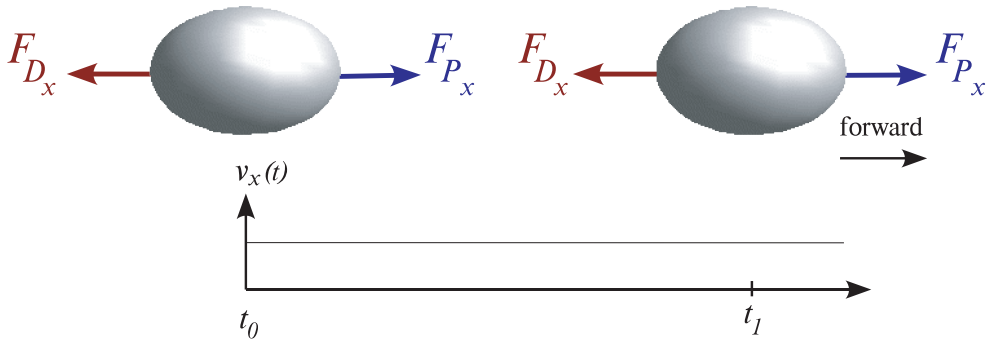


Figure 3: Identification of damping coefficients. At constant forward velocity, the propulsion force exactly counteracts the damping force.

This experiment must be run for different values of $F_{P_x}$, which can be known from Section 2.2 and the related constant velocities $v_x$ is measured using a stopwatch while the airship is traveling along a known distance. The linear and quadratic damping coefficients are determined such that the curve $v_x(F_{P_x})$ defined by Eq. 6 best fits measured data. The same method is used for the $z$-axis and symmetrical considerations imply that $D_{v_y} \tilde{=} D_{v_z}$ and $D_{v_y^2} \tilde{=} D_{v_z^2}$.

The rotational coefficients $D_{\omega_z}$ and $D_{\omega_z^2}$ are identified using a similar approach. The airship is accelerated around the yaw axis using a constant thrust until it reaches a stationary velocity, which can be measured by counting the number of revolutions per time unit.[3] Using symmetrical considerations, we also have $D_{\omega_y} \tilde{=} D_{\omega_z}$ and $D_{\omega_y^2} \tilde{=} D_{\omega_z^2}$, whereas the restoring momentum prevents from using the same method to identify $D_{\omega_x}$ and $D_{\omega_x^2}$. However, these parameters can be theoretically determined using Schlichting and Truckenbrodt [2001]:

$$D_{\omega_x} = 0 \text{ and } D_{\omega_x^2} = C_r \frac{\rho}{2} \frac{32}{15} \pi a b^2,$$

where $C_r$ typical value is in the range 0.002 and 0.006.

## 2.4 Inertia

Using the notations from Fossen [1995], the rigid-body inertia matrix can readily be written as follows:

$$\mathbf{M}_{RB} = \begin{pmatrix} m\mathbf{I}_{3\times 3} & -m\mathbf{S}(\mathbf{r}) \\ m\mathbf{S}(\mathbf{r}) & \mathbf{I}_{RB} \end{pmatrix}, \quad \text{with } \mathbf{S}(\mathbf{a}) := \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}, \tag{7}$$

where $\mathbf{I}_{3\times 3}$ is the identity matrix, $\mathbf{S}$ is the skew-symmetric matrix operator ($\mathbf{a} \in \mathbb{R}^3$) and $\mathbf{I}_{RB}$ is the inertia tensor with respect to CB. Taking into account that the axes of $\mathcal{F}_B$ correspond to the principal axes (Fig. 1) yields to a diagonal inertia tensor. The explicit description of the rigid-body inertia matrix is:

$$\mathbf{M}_{RB} = \begin{pmatrix} m & 0 & 0 & 0 & mr_z & 0 \\ 0 & m & 0 & -mr_z & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & -mr_z & 0 & I_x & 0 & 0 \\ mr_z & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z \end{pmatrix}. \tag{8}$$

[3] In order to get the rotational drag coefficients, you need to obtain a clean rotation on the spot. This can be achieved with two yaw thrusters equally distributed around the center of gravity.

This representation of inertia is however not sufficient because a bulky body in motion displaces a large amount of air particles. This phenomenon has a noticeable impact on buoyant vehicles, which have a similar density as their surrounding fluid. Therefore, the airship body experiences a resistance to its motion, which is not accounted for by the standard rigid-body inertia matrix described above. This additional effect is modeled by including added mass and inertia terms into both inertia and Coriolis matrices (Section 2.5). More precisely, the term "added-mass" (sometimes also called "virtual mass") refers to the additional inertia created by surrounding air accompanying the airship. As in Fossen [1995], we propose a simple modelling of the added-mass effect by introducing a diagonal added-mass inertia matrix[4]:

$$\mathbf{M}_A = diag(m_{A_x}, m_{A_y}, m_{A_z}, I_{A_x}, I_{A_y}, I_{A_z}). \tag{9}$$

We then derive the global inertia matrix $\mathbf{M}$ as the sum of rigid-body inertia and added-mass matrices:

$$\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A = \begin{pmatrix} m'_x & 0 & 0 & 0 & mr_z & 0 \\ 0 & m'_y & 0 & -mr_z & 0 & 0 \\ 0 & 0 & m'_z & 0 & 0 & 0 \\ 0 & -mr_z & 0 & I'_x & 0 & 0 \\ mr_z & 0 & 0 & 0 & I'_y & 0 \\ 0 & 0 & 0 & 0 & 0 & I'_z \end{pmatrix}, \tag{10}$$

where $m'_x := m + m_{A_x}$, $m'_y := m + m_{A_y}$, $m'_z := m + m_{A_z}$, $I'_x := I_x + I_{A_x}$, $I'_y := I_y + I_{A_y}$, and $I'_z := I_z + I_{A_z}$ are respectively the apparent masses and moments. Note that the shape of the envelope readily suggests that $m_{A_x} < m_{A_y} \tilde{=} m_{A_z}$, $I_{A_x} \tilde{=} 0$ and $I_{A_y} \tilde{=} I_{A_z}$ [Munk, 1934]. At this point, it could be tempting to neglect the added mass and inertia, but one should be aware that this phenomenon is responsible for an intrinsic instability of airships (see Section 2.5) and omitting it would hardly lead to realistic behaviour in simulation (as further demonstrated in the experimental part of this article).

Since the static mass is known and the distance $r_z$ has been determined in Section 2.1, the identification procedure for $\mathbf{M}$ concerns only the six diagonal elements. The inertia tensor can be calculated based on the distribution of the masses on the airship (be aware not to forget the inertia of the helium mass in the hull) and from it the entire $\mathbf{M}_{RB}$ matrix can be derived. Added-mass factors populating $\mathbf{M}_A$ can be estimated from a geometrical method based on the kinetic energy of an ideal unbounded liquid around the ellipsoid in motion. The kinetic energy and the force necessary to accelerate the airship can be computed by adding to the actual mass of the solid a fictitious mass. This added-mass is equal to the density of the fluid multiplied by a volume, which volume depends on the geometric outlines of the airship only [Munk, 1934]. These considerations result in the Lamb's $k$-factors [Lamb, 1932], where $k_1$ and $k_2$ are the inertia coefficients representing the fraction of the mass displaced by the hull (which is in turn equal to the physical mass $m$ of the airship if we assume that gravity and buoyancy are balancing each other, see Section 2.1), and $k'$ is the ratio of apparent moment of inertia to the moment of inertia of the displaced air $I_{z_h}$. In the case of an ellipsoid of revolution with semi-axes $a$ and $b$ (with $a \geq b$), this moment of inertia is given by:

$$I_{z_h} = \frac{4}{15}\pi\rho a b^2 (a^2 + b^2). \tag{11}$$

The added-mass terms can then be calculated using the Lamb's $k$-factors as follows:

$$\begin{aligned} &m_{A_x} = k_1 m \quad \text{and} \quad m_{A_y} = m_{A_z} = k_2 m, \\ &I_{A_x} = 0 \quad \text{and} \quad I_{A_y} = I_{A_z} = k' I_{z_h}. \end{aligned} \tag{12}$$

[4]This is tenable in our case since indoor blimps are moving at very low speed and we assume three planes of symmetry for the ellipsoidal hull. This reduction is also supported by the fact that off-diagonal elements are difficult to determine from experiments as well as from theory [Fossen, 1995]. Note however that in general, for higher speed and more complex shapes, coupling terms should be taken into account.

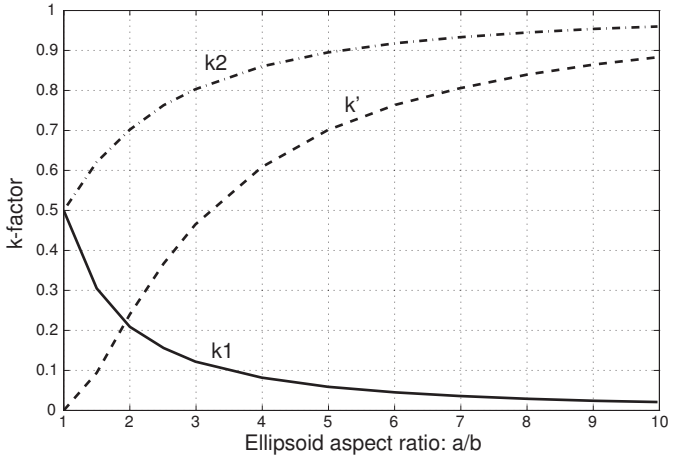


Figure 4: Lamb factors as a function of the ellipsoidal hull aspect ratio (adapted from Munk, 1936). With $k_1$ the longitudinal coefficient ($x$-axis), $k_2$ the lateral coefficient ($y$-axis) and $k'$ the rotational one (for yaw and pitch axes).

The Lamb's $k$-factors are in turn defined using two constant $\alpha_0$ and $\beta_0$:

$$\begin{aligned} k_1 &= \tfrac{\alpha_0}{2-\alpha_0}, \quad \alpha_0 = \tfrac{2(1-e^2)}{e^3}\left(\tfrac{1}{2} ln \tfrac{1+e}{1-e} - e\right), \\ k_2 &= \tfrac{\beta_0}{2-\beta_0}, \quad \beta_0 = \tfrac{1}{e^2} - \tfrac{1-e^2}{2e^3} ln \tfrac{1+e}{1-e}, \\ k' &= \tfrac{e^4(\beta_0-\alpha_0)}{(2-e^2)[2e^2-(2-e^2)(\beta_0-\alpha_0)]}, \end{aligned} \tag{13}$$

where $e$ designates the eccentricity of the ellipsoid:

$$e = \sqrt{1 - \left(\frac{b}{a}\right)^2}. \tag{14}$$

Fig. 4 displays these $k$-factors as a function of the ellipsoid aspect ratio, starting from the spherical case ($a = b$), ending with a very slender hull having the long axis up to ten times the radius ($a = 10b$). Interestingly, a spherical hull has already 50% added-mass in all directions and no additional moment of inertia ($k' = 0$). With the increase in aspect ratio, the longitudinal added-mass ($x$-axis) tend to decrease, whereas it augments in the lateral direction ($y$-axis).[5]

## 2.5 Coriolis and Centripetal Effects

Coriolis and centripetal effects are fictitious forces exerted on a body in motion when the referential frame is not inertial, which is the case for $\mathcal{F}_B$. The Coriolis force is expressed as $\boldsymbol{\omega} \times \boldsymbol{v}$ and occurs when the motion is composed of linear and rotational velocities. It accounts for the apparent force acting perpendicularly to the linear velocity vector and rotation axis, and tending to maintain the initial direction of motion without taking care of the body rotation. The centripetal force is given by $\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r})$ and is present in a rotating body when the origin of the referential frame is not CG, which is the case of $\mathcal{F}_B$. Since those fictitious forces are similar and both function of $\boldsymbol{\nu}$, they are generally put together in the form:

$$\mathbf{F}_C = \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu}, \tag{15}$$

[5] An alternative representation of added-mass terms can be found in Fossen [1995] or Khoury and Gillet [1999] and are equivalent to the one originally proposed by Lamb. Further developments for generic ellipsoids are also given in Lamb [1932] and Munk [1934].

where $\mathbf{C}(\boldsymbol{\nu})$ is the so-called Coriolis matrix. After Sagatun and Fossen [1991], it is possible to directly derive the Coriolis matrix from the inertia matrix as follows:

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{pmatrix} \mathbf{O}_{3\times3} & \mathbf{S}(\mathbf{M}_{11}\boldsymbol{v} + \mathbf{M}_{12}\boldsymbol{\omega}) \\ \mathbf{S}(\mathbf{M}_{11}\boldsymbol{v} + \mathbf{M}_{12}\boldsymbol{\omega}) & \mathbf{S}(\mathbf{M}_{21}\boldsymbol{v} + \mathbf{M}_{22}\boldsymbol{\omega}) \end{pmatrix}, \tag{16}$$

where $\mathbf{O}_{3\times3}$ is the null matrix and $\mathbf{M}_{ij}$ $(i,j = 1,2)$ are the four $3\times3$ submatrices of the global inertia matrix $\mathbf{M}$ indexed with row and column. For sake of clarity, $\mathbf{C}(\boldsymbol{\nu})$ is given explicitly in footnote[6]. Using this theorem (based on Kirchhoff's equations, Theorem 2.2 in Fossen, 1995) releases from the burden of deriving and identifying every Coriolis and centripetal terms for each of the 6 DOF. Moreover, because the inertia matrix $\mathbf{M}$ includes the added-mass terms, $\mathbf{C}(\boldsymbol{\nu})$ will include them automatically. This is of utmost importance since it explains, e.g., why an axial motion of hull shaped solid is intrinsically unstable. Any small angle between the $x$-axis and the direction of motion will indeed tend to increase [Munk, 1934]. The difference between $m'_x$ and $m'_y$ is responsible for (not cancelling) the yaw moment induced by the Coriolis effects modeled in $\mathbf{F}_C$. This resultant unstable moment is proportional to difference $(k_2 - k_1)$ of the lateral and longitudinal $k$-factors given in Eq. 13 and shown in Fig. 4. In other words, the added-mass phenomenon not only explains why the apparent inertia of the airship is higher than that predicted by $\mathbf{M}_{RB}$ but is also responsible for unintuitive behaviours such as those yawing movements occurring during forward motion. The Coriolis matrix also accounts for a number of other phenomena like slight roll inclination appearing during curved trajectories and due to the centripetal force.

# 3 Evolutionary Experiments

We evolve simple neuromorphic controllers [Nolfi and Floreano, 2000], which are asked to steer our blimp (Fig. 5) in a square room (Fig. 6) using only visual and inertial information available from on-board sensors.[7] The visual input comes from a frontal 1D camera and is preprocessed in order to feed the neuromorphic circuit with only 4 values corresponding to the contrast level in 4 parts of the field of view. Inertial information is provided by a rate gyro, which output is proportional to the rotation rate $\omega_z$ about the yaw axis.

The performance criterion (fitness function) is the mean forward velocity of the robot (measured by means of an anemometer) during a fixed period of time of 2 minutes. This fitness criterion is interesting in the sense that it does not impose a specific trajectory or behaviour, as it would be the case if an engineer would have hand-crafted the control system (see Zufferey [2005] for further discussion). It also constitutes a challenging assessment for the transfer to reality, since the evolution in simulation might well capitalize upon features of the dynamic model that are different from what happens in reality. If that were the case, evolved controllers would display lower fitness when tested on the physical robot.

## 3.1 Neuromorphic Controllers and Evolutionary Technique

The robot controller is an artificial neural network running in a tiny embedded 8-bit microcontroller (Fig. 5), which has no floating-point instructions. Activation values are restricted to integers in the range [-127,+127] and the transfer function is an hyperbolic tangent (tanh), which is stored in a lookup table in order to release the microcontroller from computing it. The connections between neurons multiply incoming activation values by a factor in the range [-7,+7]. These integer synaptic strengths are encoded on 4 bits. The whole network is genetically represented by a binary string

[6] $\mathbf{C}(\boldsymbol{\nu}) = \begin{pmatrix} 0 & 0 & 0 & 0 & -m'_z v_z & m'_y v_y - m r_z \omega_x \\ 0 & 0 & 0 & m'_z v_z & 0 & -m'_x v_x - m r_z \omega_y \\ 0 & 0 & 0 & -m'_y v_y + m r_z \omega_x & m'_x v_x + m r_z \omega_y & 0 \\ 0 & -m'_z v_z & m'_y v_y - m r_z \omega_x & 0 & -I'_z \omega_z & m r_z v_x + I'_y \omega_y \\ m'_z v_z & 0 & -m'_x v_x - m r_z \omega_y & I'_z \omega_z & 0 & m r_z v_y - I'_x \omega_x \\ -m'_y v_y + m r_z \omega_x & m'_x v_x + m r_z \omega_y & 0 & -m r_z v_x - I'_y \omega_y & -m r_z v_y + I'_x \omega_x & 0 \end{pmatrix}$

[7] In these experiment, altitude is not under evolutionary control, but is automatically regulated using distance information from the ventral infra-red sensor.

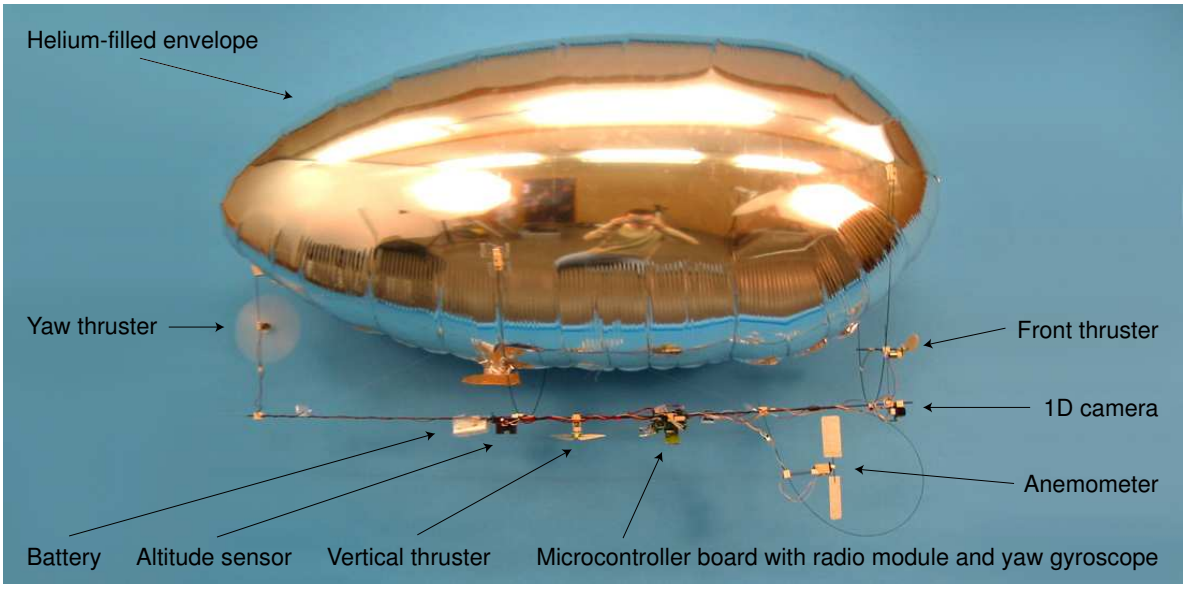


Figure 5: The physical flying platform, named Blimp2b. The envelope measures 110x60x60cm and provides a lift capacity of about 200g. The underneath gondola is made of thin carbon rods supporting several electronic components: three thrusters (8mm DC motors, gear and propellers from DIDEL$^{TM}$), a forward-looking 1D camera (Taos inc. TSL3301) with 50 active pixels covering an horizontal field of view of 70°, an anemometer (free rotating propeller mounted on a shaft with an optical encoder whose rotation rate is roughly proportional to the forward velocity of the blimp, developed by DIDEL$^{TM}$), a vertical distance sensor (Sharp$^{TM}$ GP2Y0A02YK), a MEMS piezoelectric rate gyro (AnalogDevices$^{TM}$ ADXRS300) measuring yaw rotation, and an electronic board featuring an 8-bit microcontroller running at 20MHz (Microchip$^{TM}$ PIC18F452) together with a Bluetooth module (Mitsumi$^{TM}$ WML-C10-AHR) for bidirectional wireless communication with a ground station. On-board energy is provided by a 1200mAh Lithium-polymer battery, which is sufficient for 2-3 hours autonomy. Further details on the microcontroller board, wireless link, and camera can be found in Zufferey et al. [2003].

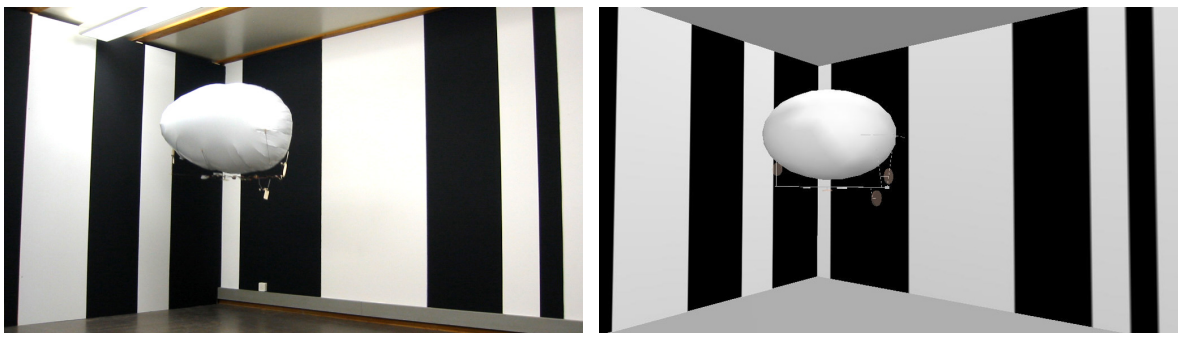

Figure 6: Real (left) and simulated (right) robots and environments. The square room measures 5x5m by 3m high and features randomly contrasted walls. The patterns in the simulator are exactly reproduced from the real ones that have been painted on the walls. Note that spatial frequency of pattern can vary substantially among the walls. The simulated Blimp2b has the same sensors as the real one: 1D camera, anemometer and yaw rate gyro, and vertical distance sensor. Sensor values are disturbed with noise as observed from measurements with the real sensors.

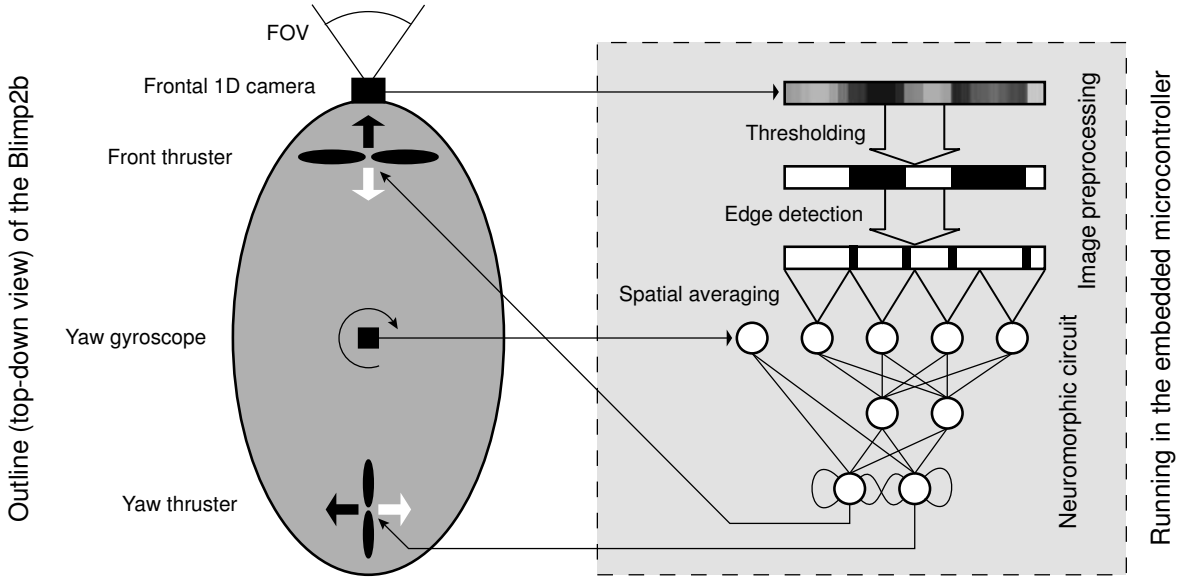


Figure 7: Left: Outline of the blimp sensory inputs and actuators. Right: Neural network architecture and input preprocessing.

composed of a series of 4-bit blocks, each corresponding to one synapse. In these experiments, only the synaptic strengths are evolved. The neural network (Fig. 7) has 4 input units receiving visual information, 1 input neuron connected to the rate gyro, 1 bias unit (not shown), 2 hidden units, and 2 output neurons driving the frontal and yaw thrusters. Output neurons have recurrent and lateral connections (with a fixed time delay of 1 sensory-motor cycle).

A population of 60 individuals (corresponding to 60 88-bit genetic strings) is evolved using rank-based truncated selection, one-point crossover, bit mutation, and elitism [Nolfi and Floreano, 2000]. The genetic strings of the first generation are initialised randomly. After ranking the individuals according to their measured fitness values, the top 15 individuals produce 4 copies each to create a new population of the same size and are randomly paired for crossover. One-point crossover is applied to each pair with probability 0.1 and each individual is then mutated by switching the value of a bit with probability 0.05 per bit. Finally, a randomly selected individual is substituted by the original copy of the best individual of the previous generation (elitism).

Each individual of the population is evaluated on the robot two times for 1200 sensory-motor cycles (corresponding to 2 minutes real time). After each evaluation period, a backward movement of 4 seconds with random yaw commands is executed to create a . initial situation for the next evaluation. The behaviour of an individual is evaluated by means of the anemometer (Fig. 5), whose rotation speed to the left is proportional to forward displacement of the blimp. The fitness is thus computed as the average forward velocity over the entire evaluation period of the individual. In contrast to previous experiments entirely run in reality [Zufferey et al., 2002], simulation enables to put additional pressure against behaviours that are not desirable. In order to evolve not only efficient navigation in the sense of high average forward velocity, but also reliable collision avoidance, a series of virtual proximity sensors have been implemented all around the hull of the simulated robot. Whenever one of these sensors is active (i.e., when the hull is closer than 25cm from a wall), the current velocity indicated by the anemometer is forced to zero. In addition, individuals that display poor behaviours (low fitness) are interrupted prematurely (100 cycles) in order to reduce simulation time. This procedure encourages evolution to select individuals that are not only able to move forward at high velocity, but also to quickly escape from difficult initial positions such as, e.g., facing a wall.

## 3.2 Evolution Management and Simulator Implementation

To run our evolutionary experiments, we employ a framework consisting of two pieces of software. The first one, *goevo*[8], is custom software that manages evolution, creates neuromorphic circuits, provides sensor display facilities, and can connect to a range of physical or simulated robots, included the Blimp2b via a wireless connection (using Bluetooth) or its simulated version over TCP/IP. The second is the simulator itself, Webots$^{TM}$ [Michel, 2004], which is a convenient program allowing to create and run mobile robot simulations in 3D environments (based on OpenGL) with a number of built-in sensors like bumpers, range finders, or cameras. Fig. 6 illustrates the Blimp2b in its real environment (on the left) and its simulated counterpart in Webots (on the right).

Webots also features rigid-body dynamics (based on ODE[9]). This dynamics engine provides libraries for kinematics transformations, collision handling, friction and bouncing forces, etc. However, it does not yet support non-rigid-body effects such as aerodynamic or added-mass effects. Therefore, we implemented the dynamic model of our blimp as custom library, while leaving to ODE the emulation of friction and bouncing forces. The dynamics implementation takes the current orientation and velocities as input and provides force vectors that are passed to ODE, which computes the resulting new state after a simulation step. [10]

In order to make sure that the simulated behaviour is as close as possible to reality, we carefully followed the parameter identification procedure described above. Sensors were modeled using experimentally recorded data. Since it has been shown that appropriate levels of noise in sensor models is a prerequisite to facilitate the transfer to reality [Jakobi et al., 1995], noise level and envelope is reproduced in the sensor model so to match as closely as possible real data.

The simulation rate with all sensors enabled and full 3D physics (ODE and custom blimp model) is 40 to 50 times faster than real-time when running on current computers (e.g., Intel(R) Pentium IV at 2.5GHz with 512MB RAM and nVidia(R) GeForce4 graphic accelerator).

## 3.3 Results

### 3.3.1 Evolution in Simulation

We performed five evolutionary runs, each starting with a different random initialization (Fig. 8.A). The best evolved individuals of the five runs developed efficient behaviours in less than 50 generations to navigate in the room in the forward direction while actively avoiding walls. Fig. 8.B illustrates the typical preferred behaviour of the best evolved individual. The resulting circular trajectory is close to optimal because it well fits the available space (the back of the blimp sometimes gets very close to a wall without touching it). It is to notice however that the trajectory is not centered in the room and this is probably due to the spatial frequency discrepancy between walls (top and right sides have less vertical stripes than the two others). The non-zero angle between the heading direction of the blimp (indicated by the small segments) and its trajectory indicates that the simulated flying robot is always side-slipping and thus evolved controllers have to take into account the quite complex dynamics by relying on air drag to compensate the centripetal force.

In order to further assess the wall-avoidance capability of the evolved robots, we artificially reduced the size of the room and tested the same best individual in the tighter environment. As a result, the blimp modified its trajectory into a more elliptic one (Fig. 8.C). In another test, we deliberately positioned the best individual frontally against a wall (Fig. 8.D). In this situation, it actively reversed its front thruster, backing away from the wall while rotating about its yaw axis in order to recover its preferred circular trajectory and accumulate fitness points.

[8] Goevo website: http://lis.epfl.ch/resources/evo/

[9] Open Dynamics Engine: http://opende.sourceforge.net

[10] The model of our Blimp2b is freely available under GPL in the current distribution of Webots and can be downloaded from our website as well: http://lis.epfl.ch/software.php

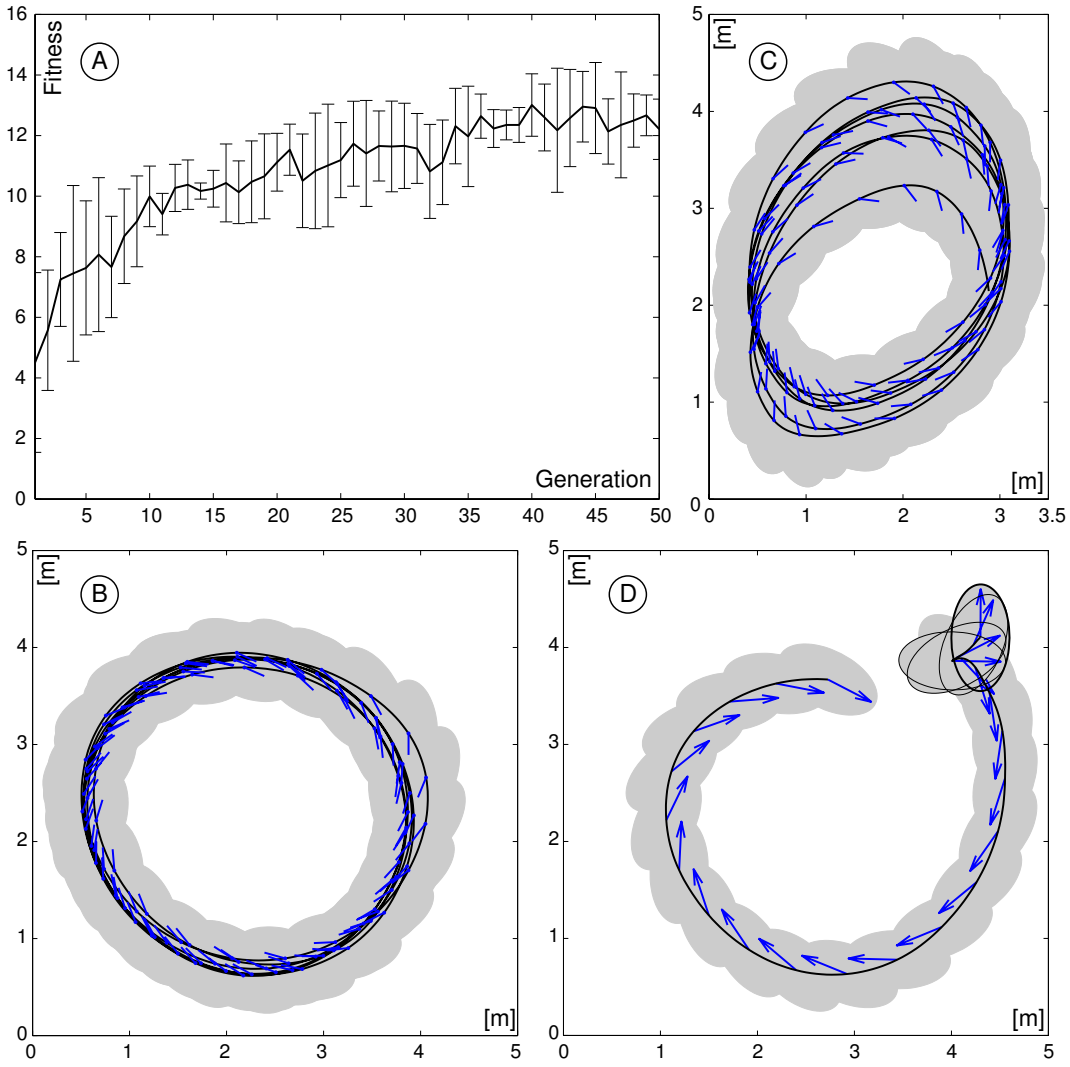


Figure 8: Results in simulation. A) Average fitness values and standard deviations, over a set of five evolutionary runs, of the fittest individuals on each evaluation. B) Top-down view of the typical trajectory of the fittest evolved individual during 1200 sensory-motor cycles. The black continuous line is the trajectory plotted with a time resolution of 100ms. The small segments indicate the heading direction every second. Light-gray ellipses represent the envelope of the blimp also plotted every second. C) Trajectory of the fittest individual when tested during 1200 sensory-motor cycles in a room that has been artificially shrunk by 1.5m. D) When the same best individual is started against a wall, it first reverses its front thruster while quickly rotating clockwise before resuming its preferred behaviour. The ellipse surrounded with bold black line indicates the starting position. The following ones with black outline indicates the envelope when the robot is in backward motion. Arrows indicate frontal direction irrespective of forward or backward movement.

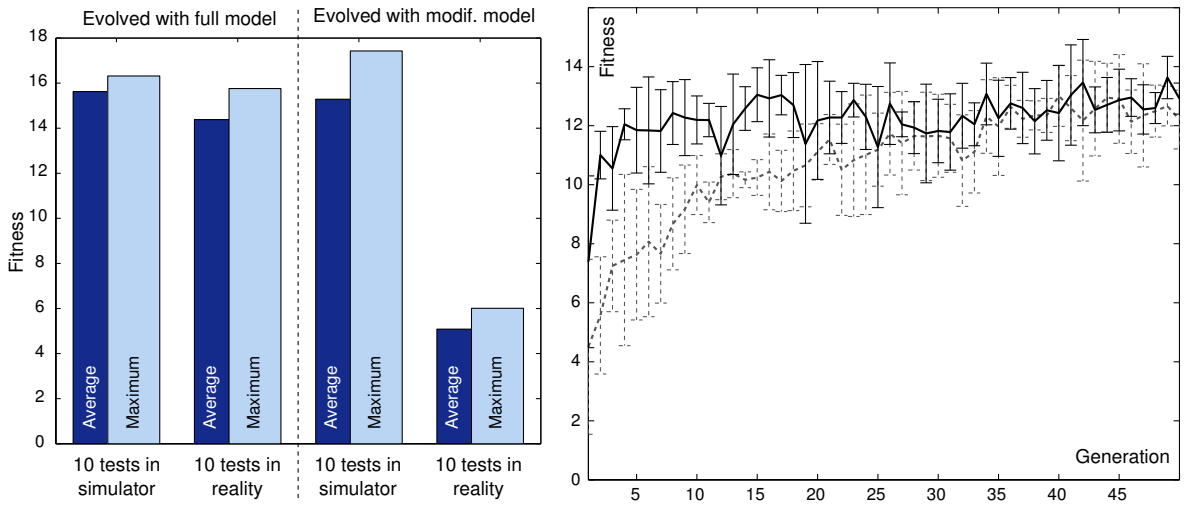


Figure 9: Left: Navigational performance: from simulation to reality. Fitness results of 10 trials of best evolved individual, once when evolution is carried out with the full dynamics model, and once when the model has been simplified (no Coriolis effects due to added-mass). Right: Fitness graph of the evolution in simulation with the simplified dynamics model. The dashed line is the fitness graph replotted from Fig. 8.A, for sake of comparison.

### 3.3.2 Transfer to Reality

A series of experiments were carried out in order to evaluate the quality of the transfer from simulation to reality. The first one consisted of testing in reality the fittest controller evolved in simulation. When directly transferred to the real airship, without further evolution, the physical robot behaviour was very similar to the simulated one.[11] After random initialization in its environment, the robot was able to quickly drive itself on its preferred circular trajectory, while reliably avoiding contact with surrounding walls. We measured the transfer performance based on a slightly modified version of the fitness function. Since no proximity sensors were mounted on the physical robot, we simply cancelled their effect on the fitness function. The fittest individual was tested 10 times in simulation and 10 times in reality during 1200 sensory-motor cycles (2 minutes in reality). The results of these tests (left side of the histogram in Fig. 9) show that controllers evolved in simulation obtain very similar performance in reality.

A second experiment was carried out in order to check the necessity of a realistic dynamic model. To this aim, we run a set of five evolutionary experiments in simulation with a modified equation of motion (Eq. 1) where the added-mass Coriolis effects were cancelled. This modification eliminates the spontaneous rotational effects described in Section 2.5, making the piloting of the simulated blimp easier, but less realistic. The final fitness values obtained in simulation (Fig. 9, right) were very similar to those obtained with the full model. However, when the fittest controller was transferred to the real robot, the performance significantly dropped (Fig. 9, right side of the histogram), attesting for the necessity of a realistic dynamic model.

In order to further check the correspondence between results with the full model and the real robot, we run a third experiment where we compared the signals from the anemometer, rate gyro and actuators while the blimp was backing away from a frontal wall, as shown in Fig. 8.D, both in simulation and in reality. Fig. 10 shows the very close match between signals gathered in reality and those recorded in the same situation in simulation. At the beginning, the front thruster is almost fully reversed while a strong yaw torque is produced by the yaw thruster. These actions produce the

[11] Video clips of simulated and physical robots under control of the same evolved neural controller are available at http://lis.epfl.ch/jczthesis

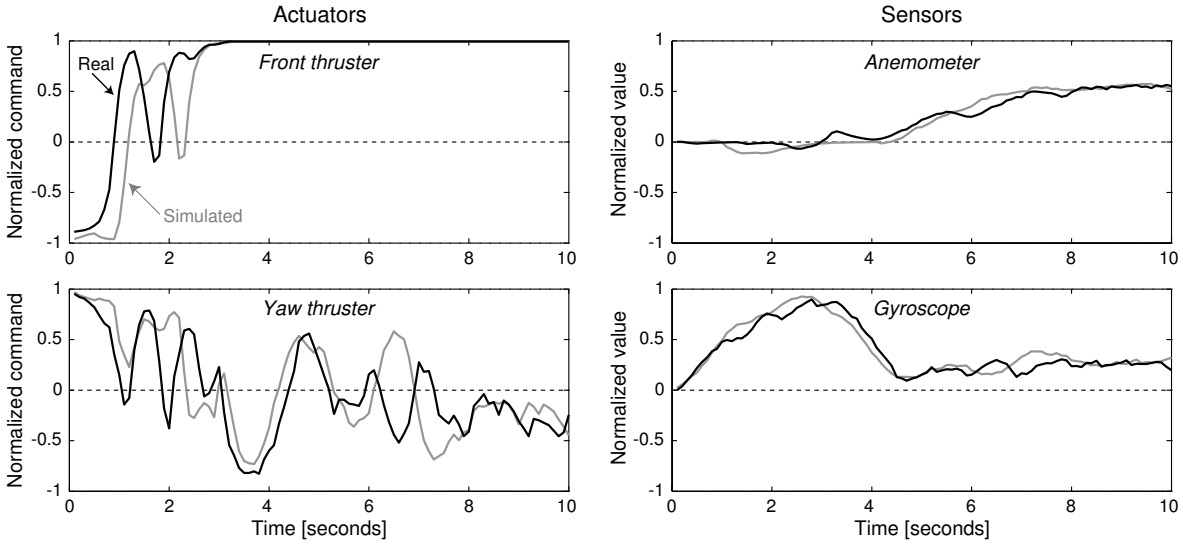


Figure 10: Comparison of thruster commands and sensor values in simulation and reality when the best evolved individual is started facing a wall, as shown in Fig. 8.D.

same increase in rotation rate (detected by the rate gyro) and a slight backward velocity (indicated by negative anemometer value), both in reality and in simulation. After approximately 3 seconds, the blimp is almost finished with the back-and-rotate manoeuvre and starts a strong counter-action with the yaw thruster to cancel the rotational movement, resulting in a noticeable decrease in gyro output. After that, the robot accelerates forward (as shown in the anemometer graph) to recover its preferred circular trajectory (as revealed by the almost constant, though not null, rate gyro output). Slight discrepancies among signals from simulation and reality can be explained by differences in initial positions implying slightly different visual inputs, inaccuracies in sensor modelling, and omitted higher order components in the dynamic model.

# 4 Conclusions and Outlook

An exhaustive dynamic modelling of an indoor airship has been presented together with a pragmatic methodology for parameter identification using a series of simple experiments to be carried out with the physical blimp. Following this methodology allowed us to develop an efficient flight simulator without the need for costly facilities like wind tunnel or aerodynamic balance. The obtained 3D simulation runs 40 to 50 times faster than real-time, significantly speeding up evolutionary experiments, which are notoriously time-consuming.

A major drawback of using artificial evolution to discover efficient robot controllers, that is the time it takes for evaluating a whole population of controllers over a number of generations (which is even more problematic with physical flying robots that may be damaged by collisions, have limited on-board energy, and have actuators performance that change over time), has been addressed in this article by demonstrating the use of physics-based simulation and successful transfer to reality. Not only qualitatively identical behaviours have been observed, but a close look at the internal signals reveal a quantitatively similar functioning of the evolved control system during the same phase of operation in the two worlds, simulated and real.

However, systematic analysis of the importance of each parameter and component of the model has not been carried out and good correlation between simulation and reality has been demonstrated only within a specific navigational task where altitude was automatically kept constant. In the future,

care should be taken to assess whether the vertical and roll movements are sufficiently close to the real dynamics in order to ensure a good transfer in the case of more complex tasks with more complex sensors.

So far, the vision sensor as well as the visual surroundings remained quite simplistic. In future work, we plan to take advantage of this simulator to evolve more complex neuromorphic systems (requiring a higher number of generation to produce efficient solutions) fed with a higher number of pixels in order to cope with natural indoor environments. In that case, it is probable that the visual input will have significant discrepancies between simulation and reality, such that hand-crafted image preprocessing will fail at totally cancelling them. In order to cope with that kind of issues, previous research in evolutionary robotics suggest two approaches. The first one is called incremental evolution and consists in continuing evolution in reality for a short amount of generations (see Nolfi and Floreano, 2000, Section 4.4). The second solution consists of evolving hebbian-like synaptic plasticity, which we have shown to support fast self-adaptation to changing environments [Urzelai and Floreano, 2001].

Another interesting research direction that is enabled by working in simulation is the evolution of sensor morphologies. For instance, position and orientation of simple vision sensors could be left to evolutionary control [Cliff and Miller, 1996]. Eventually, our goal is to apply this approach to winged microflyers [Nicoud and Zufferey, 2002, Zufferey and Floreano, 2006, Zufferey et al., 2006] instead of airships.

**Acknowledgements**

The authors are grateful to Olivier Michel for his support related to Webots, to the *goevo* developers team, to Claudio Mattiussi for useful discussions on the mathematics, and to Jean-Daniel Nicoud for providing parts and support for building the Blimp2b. The authors are equally indebted to the anonymous reviewers, whose constructive comments have allowed them to improve this article. This work has been supported by the Swiss National Science Foundation.

# References

S. Bermúdez i Badia, P. Pyk, and P. Verschure. A biologically based flight control system for a blimp-based uav. In *IEEE International Conference on Robotics and Automation*, April 2005.

D. Cliff and G.F. Miller. Co-evolution of pursuit and evasion ii: Simulation methods and results. In P. Maes, M. Mataric, J.A. Meyer, J. Pollack, H. Roitblat, and S. Wilson, editors, *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press-Bradford Books, 1996.

F.M. da Silva Metelo and L.R. Garcia Campos. Vision based control of an autonomous blimp. Technical report, 2003.

T.I. Fossen. *Guidance and Control of Ocean Vehicles*. Wiley, New York, 1995.

F. Iida. Biologically inspired visual odometer for navigation of a flying robot. *Robotics and Autonomous Systems*, 44:201–208, 2003.

N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. *Lecture Notes in Computer Science*, 929:704–720, 1995.

G.A. Khoury and J.D. Gillet. *Airship Technology*. Cambridge University Press, London, 1999.

H. Lamb. *Hydrodynamics*. Cambridge University Press, London, 1932.

C. Melhuish and J. Welsby. Gradient ascent with a group of minimalist real robots: Implementing secondary swarming. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2002.

O. Michel. Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

M.M. Munk. Fluid mechanics, second part. In W.F. Durand, editor, *Aerodynamic Theory I*, pages 224–304. Julius Springer, 1934.

M.M. Munk. Aerodynamics of airships. In W.F. Durand, editor, *Aerodynamic Theory VI*, pages 32–48. Julius Springer, 1936.

J.D. Nicoud and J.-C. Zufferey. Toward indoor flying robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, pages 787–792, 2002.

S. Nolfi and D. Floreano. *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines.* MIT Press, Cambridge, MA, 2000.

C. Planta, J Conradt, A. Jencik, and P. Verschure. A neural model of the fly visual system applied to navigational tasks. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2002.

S.I. Sagatun and T. Fossen. Lagrangian formulation of underwater vehicles' dynamics. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1029–1034, 1991.

H. Schlichting and E. Truckenbrodt. *Aerodynamik des Flugzeuges*. Springer, Berlin, 2001.

J. Urzelai and D. Floreano. Evolution of adaptive synapses: robots with fast adaptive behavior in new environments. *Evolutionary Computation*, 9:495–524, 2001.

S. van der Zwaan, A. Bernardino, and J. Santos-Victor. Visual station keeping for floating robots in unstructured environments. *Robotics and Autonomous Systems*, 39:145–155, 2002.

H. Zhang and J. Ostrowski. Visual servoing with dynamics: Control of an unmanned blimp. Technical report, 1998.

J.-C. Zufferey. *Bio-inspired Vision-based Flying Robots*. PhD thesis, Swiss Federal Institute of Technology in Lausanne (EPFL), 2005.

J.-C. Zufferey and D. Floreano. Fly-inspired visual steering of an ultralight indoor aircraft. *IEEE Transactions on Robotics*, 22(1):137–146, 2006.

J.-C. Zufferey, D. Floreano, M. van Leeuwen, and T. Merenda. Evolving vision-based flying robots. In Bülthoff, Lee, Poggio, and Wallraven, editors, *Biologically Motivated Computer Vision: Second International Workshop, BMCV 2002, Tübingen, Germany*, volume 2525 of *Lecture Notes in Computer Science*, pages 592–600. Springer-Verlag, 2002.

J.-C. Zufferey, A. Beyeler, and D. Floreano. Vision-based navigation from wheels to wings. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, volume 3, pages 2968– 2973, 2003.

J.-C. Zufferey, A. Klaptocz, A. Beyeler, J.D. Nicoud, and D. Floreano. A 10-gram microflyer for vision-based indoor navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, 2006.