A Contribution to Vision-Based Autonomous Helicopter Flight in Urban Environments

Laurent Muratet[a] Stephane Doncieux[a] Yves Briere[b]
Jean-Arcady Meyer[a]

[a]*AnimatLab - LIP6*
8, rue du Capitaine Scott
75015 Paris, France
http:animatlab.lip6.fr
muratet@ia.lip6.fr

[b]*ENSICA*
1, Place Emile Blouin
31056 Toulouse Cedex 5, France
http://www.ensica.fr
yves.briere@ensica.fr

Abstract

A navigation strategy that exploits the optic flow and inertial information to continuously avoid collisions with both lateral and frontal obstacles has been used to control a simulated helicopter flying autonomously in a textured urban environment. Experimental results demonstrate that the corresponding controller generates cautious behavior, whereby the helicopter tends to stay in the middle of narrow corridors, while its forward velocity is automatically reduced when the obstacle density increases. When confronted with a frontal obstacle, the controller is also able to generate a tight U-turn that ensures the UAV's survival. The paper provides comparisons with related work, and discusses the applicability of the approach to real platforms.

Key words: Helicopter, optic flow, obstacle-avoidance, urban environment

Airborne devices are specific platforms whose control raises distinctive difficulties as compared to ground robots. For instance, they can hardly rely upon usual sensors to navigate, especially if the challenge is to let them move in an urban environment – because infra-red sensors are sensitive to external light and can only detect nearby obstacles, because sonar sensors are too slow or too heavy for small platforms, and because lasers are too dangerous to be used in the presence of unadvised humans. Not surprisingly, under these conditions, there is currently no obvious solution to the problem of coming up with sensors to equip a small UAV[1] capable of flying in an urban environment without hitting obstacles. This is why, in this paper, we draw inspiration from biology to assess the potential for using vision to tackle this problem. Indeed, because the distance between their two eyes can be very small in many animal species, such individuals cannot rely on stereoscopic vision to assess the distances of surrounding objects [10]. They, therefore, have developed specific neural processes based on motion detection and *optic flow* monitoring to estimate these distances. In this paper, we have implemented such a biomimetic strategy on a simulated helicopter moving in a realistic 3D urban environment to assess its adaptive value under these conditions.

The results described below are based on a realistic physical model of a rotary-wing UAV which is combined with a 3D-engine generating images that are fed into an obstacle-avoidance system. This system interacts with a low-level controller responsible for keeping the helicopter as stable as possible. The 3D environment is generated and monitored with Crystal Space[2], an open-source software. A real-time correlation-based algorithm is used to compute the optic flow.

The paper successively describes the physical model of the UAV and its low-level controller, the optic flow extraction algorithm, the navigation strategy and its biological basis, and the high-level controller for obstacle-avoidance. The results of several simulations performed in three different urban environments are then presented and discussed. Finally, possible improvements to the system are suggested.

1 The simulated UAV platform

1.1 The physical model

The simulated robot is a rotary-wing UAV inspired from the Concept 60 SR II (Fig. 1), a remote-controlled helicopter produced by Kyosho[3]. It has six degrees of freedom: three coordinates (x, y, z) and three attitude angles, i.e., the yaw (ψ), the pitch (θ) and the roll (ϕ). It weighs 4.5 kg and is 140 cm long, 15 cm wide, and 47 cm high. Its main rotor has a diameter of 150 cm, while that of the tail rotor is 26 cm. It is assumed to be able to carry the visual system described below. To simulate this device, we use Autopilot, a physical

[1] Unmanned Aerial Vehicle
[2] http://crystal.sourceforge.net
[3] http://www.alansmodels.com/helis/con60sr2.htm

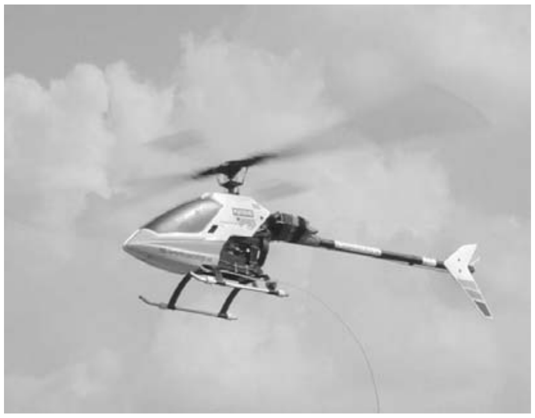

Fig. 1. The Concept 60 SR II.

simulator that has already been used efficiently for such a purpose[4].

1.2 The robot's sensors and environment

The navigation strategy implemented in this work calls upon two sensors: a video camera and an Inertial and Attitude Measurement Unit (IAMU) supposedly equipped with Kalman-filtered accelerometers, gyrometers and magnetometers like those readily available off the shelf[5]. To simulate the remaining measurement errors, the altitude, attitude and speed input to the UAV's controllers are corrupted by a Gaussian white noise as described below.

1.3 The robot's low-level controller

A helicopter is an unstable platform that must be permanently controlled. Therefore, in addition to using optic flow to monitor horizontal displacements that allow obstacle avoidance, the robot's six degrees of freedom must be simultaneously controlled. A low-level controller has been designed for this purpose enabling straight-line flight, with a small lateral component V_y and at a constant altitude (Fig. 2). A linear model of the helicopter has been identified around a nominal state (straight flight at low speed). A coupling between M (main rotor collective) and ω_z (heading rotational speed) has been compensated by a simple feedforward gain. With this simple pre-control, the helicopter's degrees of freedom are well decoupled, thus affording an efficient design of four independent PID controllers. The main rotor collective M is used to maintain the helicopter at a constant altitude z. During turns, the

[4] http://autopilot.sourceforge.net/gallery.html

[5] see, e.g., http://www.micropilot.com, http://www.xsens.com, or http://www.microstrain.com

tail rotor collective T controls the yaw ψ. The longitudinal attitude block regulates pitch θ and consequently forward motion V_x. The lateral attitude block controls roll ϕ and lateral motion V_y that the strategy described below needs to hold to a minimum. The values of $\psi, \theta, \phi, V_x, V_y$ and z are obtained from the internal states of the simulator. The experimental results described below demonstrate that this low-level controller provides a high close-loop bandwidth and affords efficient capacities for perturbation resistance.

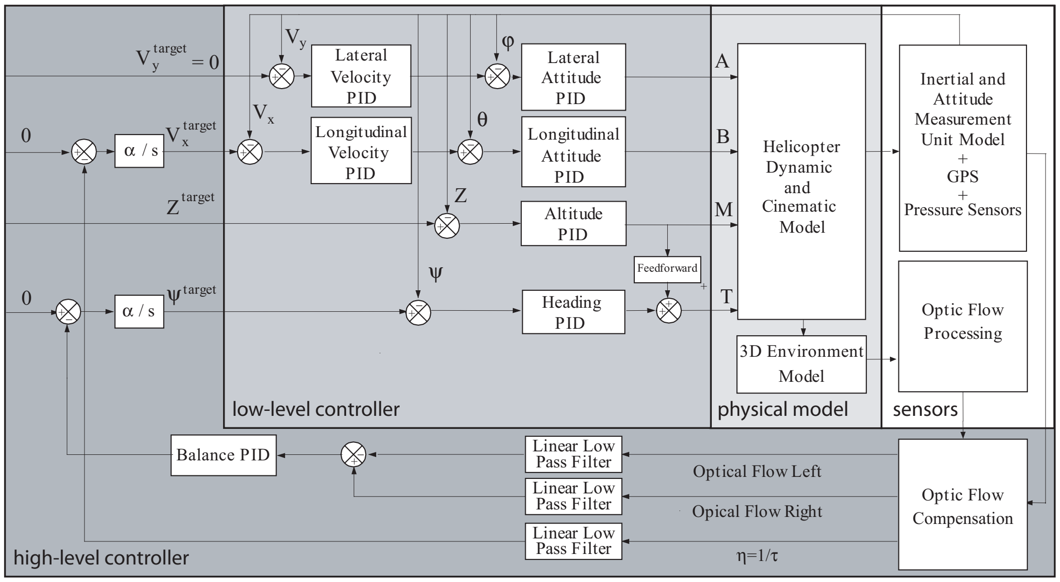

Fig. 2. The helicopter's low-level controller calling upon six PID modules. Z^{target} remains constant and V_y^{target} remains null during an experiment. The values of ψ^{target} and V_x^{target} are determined by the high-level obstacle-avoidance controller. The dynamics of the helicopter depend upon four output signals: A, B, M and T. The high-level controller calling upon optic flow calculation is described in Section 3.

2 Optic flow

2.1 Introduction

When the external world is projected onto the image plane of a camera, the movements of each point in this plane define the so-called *motion field*, from which information about the self-motion of the camera or about the structure of the scene can be inferred. When such movements are sampled, by means of a video stream for instance, the apparent motion of pixels in the image constitutes the optic flow, which is a convenient approximation of the motion field if the intensity of each pixel is preserved from one frame to the next.

This is why, to compute the optic flow in this work, the hue of each pixel (i, j) is used to evaluate its intensity value $I(i, j)$, because this type of data is not very sensitive to changes in light intensity.

Several varieties of algorithm can be used to compute optic flow [3]. Differential methods [21,26] call upon spatio-temporal intensity derivatives. Correlation approaches [1] rely on feature matching. Frequency-based methods [16] use velocity-tuned filters in the Fourier domain. In this work, correlation techniques have been preferred because they are known to be robust and because experience demonstrates that they are accurate enough to make efficient control possible.

2.2 Region matching computational principle

Standard correlation-based methods try to determine, for each pixel (i, j) in an image, the $v \times v$ pixel-wide patch P_v centered on $(i+u, j+w)$ in frame t that best matches the patch P_v centered on (i, j) in frame $t-1$. Matching patches are sought in a square region delimited by points of coordinates $(i-n, j-n)$ and $(i+n, j+n)$. For each possible match, a matching-distance M is computed from Eq. (1) that compares the pixel intensities I of the two patches.

For each pixel (i, j): $\forall (u, w) \in \{-n, \ldots, n\} \times \{-n, \ldots, n\}$

$$M(i, j; u, w) = \sum_{(x,y) \in P_v} \Big| I_{t-1}(i+x, j+y) - I_t(i+x+u, j+y+w) \Big| \tag{1}$$

Under these conditions, the actual motion of pixel (i, j) is (u, w), which corresponds to the displacement between (i, j) and the centre of the patch that minimizes the matching-distance M (Fig. 3). However, to determine this motion is computationally expensive and hardly applicable to real-time applications because of the quadratic dependency of the search upon n, the maximum motion detected. To find the best one, the algorithm must compute $(2n+1) \times (2n+1)$ matches. This is why we used a variant of this algorithm - the so-called *real-time quantized* version [7,8] - that was more compatible with our needs.

2.3 Real-time application

The real-time quantized algorithm is a correlation-based algorithm in which the search over space is replaced by a search over time. Using Eq. 1, the matching-distance is computed in a $n = 1$ neighbourhood. Thus, the possible values for u and w are restricted to the range $\{-1, 0, 1\}$. For each pixel, a

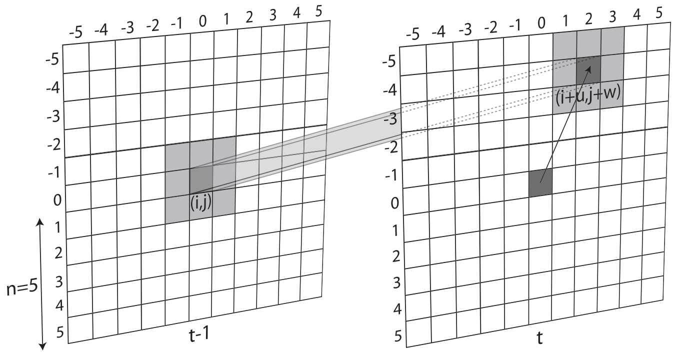

Fig. 3. Best match between two P_v patches of pixels in two consecutive frames. Accordingly, pixel $(0, 0)$ is assumed to have moved to position $(2, -4)$ between frames $t - 1$ and t. In this example, 3×3 pixel-wide patches, and a target region defined by a n value of 5, were used.

motion between the image at time $t - 1$ and the image at time t is computed, as well as motions between images at $t - 2$ and t, $t - 3$ and t, and so forth until $t - D$, D being an empirically-determined depth value.

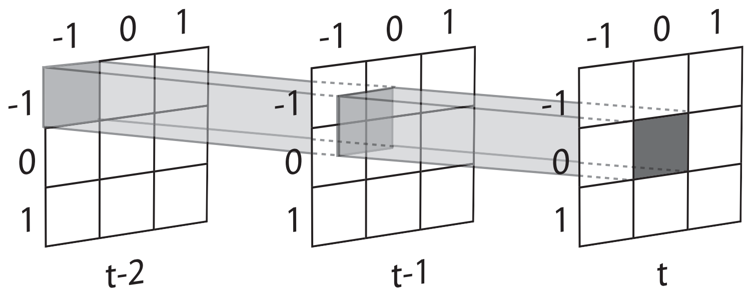

Fig. 4. A motion of $(1, 1)$ between $t - 2$ and t is equivalent to a motion of $(\frac{1}{2}, \frac{1}{2})$ between $t - 1$ and t, if a constant motion is assumed.

As demonstrated on Fig. 4, assuming constant motion over D frames, the actual motion of pixel (i, j) is $(\frac{u}{i}, \frac{w}{i})$, which corresponds to the displacement (u, w) between (i, j) in frame $t - i$ and the centre of the patch in frame t that minimizes the matching-distance M. As a consequence, this variant evaluates pixel motions in the range $\frac{1}{D}$ to 1, while the original algorithm operates in the range 1 to n. Moreover, with n set to 1, the variant computes 3×3 matches per frame, i.e., $9 \times D$ matches for D frames. Therefore, this method is a trade-off between a quadratic search over space and a linear search over time. There is, however, a minor drawback associated with this computational gain, because the method doesn't allow motions greater than one pixel per frame to be detected. Nevertheless, this limitation may be overcome by considering blocks of pixels instead of isolated ones. Pooling pixels by blocks of $p \times p$ multiplies the amplitude of detectable motions by p, and reduces the number of pixels considered by $p \times p$. Hence, it improves both the range and the speed of the algorithm. Doing this, the range goes from $[\frac{1}{D}, 1]$ to $[\frac{p}{D}, p]$.

The optic flow thus computed is a rough estimate of the motion field, as it only detects motions of D different amplitudes in eight directions. Nevertheless, this method entails only simple computations and proves to be less time-consuming and more robust than many other methods. Cobos and Monasterio[9] implemented the corresponding algorithm on a FPGA, thus making the real-time processing (25 images/second) of the optic flow on 100x100 pixel images possible.

3 Obstacle-avoidance

3.1 Biological inspiration

Flying insects, such as houseflies or bees, have sensory-motor abilities that still outperform those of UAVs. Beyond their flying manoeuvrability, they benefit from a reactive navigation system that enables them to wander in cluttered environments thanks to on-line monitoring of the optic flow. In 1865, Von Helmholtz explained how these animals are able to evaluate the distances of lateral objects by using motion parallax and, since that time, numerous studies have investigated the corresponding mechanisms. Srinivasan et al. [37], for instance, have studied how bees call upon a *centering response*, which consists in equalizing the optic flow on left and right sides, to fly in the middle of a textured tunnel. Bees also exhibit a *clutter response* that allows their speed to be adapted to the width of the tunnel by maintaining a constant average motion [38]. Srinivasan et al. implemented the corresponding strategies on a wheeled robot to demonstrate their efficiency [36]. Likewise, Franceschini et al. [18] described how the organization of the compound eye of the housefly, and how the neural processing of visual information obtained during the flight, allow this insect to compute its distances to lateral obstacles and to avoid them. This biological knowledge was exploited to implement opto-electronic devices on several flying robots [28,30,39,17,31].

It also appears that other scientists have understood how other animals use the frontal optic flow to estimate the so-called *time-to-contact*, i.e., the time before a frontal collision is likely to occur. The gannet is supposed to use this information when it quickly dives to catch a fish and determines the exact moment when to fold its wings before entering the water [23–25]. Camus [6] has implemented the corresponding strategy on a wheeled robot.

In the following, we use the helicopter's body-frame to refer to a point (x, y, z) in the environment and the classical image-frame to refer to a pixel (i, j). The *Focus Of Expansion* (FOE), which is the projection of the direction of the helicopter onto the image, is taken as the center of the image frame. Parameter f refers to the focal length of the camera. We will use symbols $\overrightarrow{M}$ and $\overrightarrow{V}$, respectively, to distinguish between motion on the image plane and velocity in the environment.

The component of the optic flow that is required to avoid obstacles is generated by a forward translation V_x of the observer. In this case, the horizontal motion M_h of an object is proportional to the inverse of its distance to the observer. From Fig. 5, the following relationship may be derived:

$$\frac{h}{Y} = \frac{f}{X} \tag{2}$$

As the low-level controller prevents the helicopter from skidding aside, we make the assumption than the lateral velocity V_y is null. After differentiating with respect to time, the above equation becomes:

$$\dot{h} = -f.Y.\Big(\frac{\dot{X}}{X^2}\Big) \tag{3}$$

Substituing $\frac{h.X}{f}$ for Y and replacing $\dot{X}$ by $-V_x$ and $\dot{h}$ by M_h, one obtains:

$$\frac{M_h}{h} = \frac{V_x}{X} \tag{4}$$

Replacing X by $d.\cos\beta$, where β is the relative angle of the perceived object, it finally appears that horizontal motion M_h of pixels is given by the the equation:

$$M_h = \frac{h.V_x}{d.\cos\beta} \tag{5}$$

From Eq. (5), we can deduce that a strategy equalizing the perceived pixel motions will tend to maintain equal the distances to obstacles on both sides of the helicopter. This strategy was called either the *balance strategy* in [12,13,11,14] or the *centering response* in [38,4].

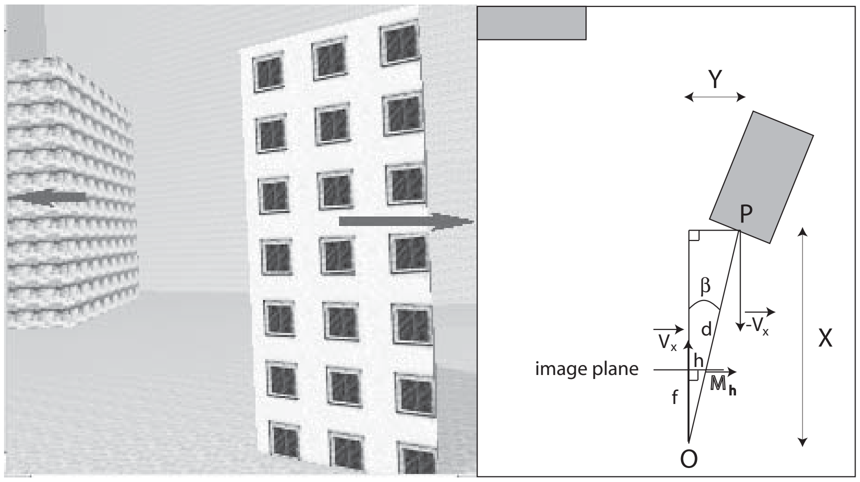

Fig. 5. The helicopter in point O perceives obstacle P at a distance d with a relative angle β. During a forward translation, any object in the environment is perceived to be moving away from the center of the image (left figure). The obstacle on the right being closer than the one on the left, its horizontal apparent motion is greater and the helicopter must turn left.

According to Eq. (6) below:

$$\Delta\psi^{target} = k_{psi}.\left(\overline{M_h^{right}} - \overline{M_h^{left}}\right) \tag{6}$$

in which $\overline{M_h^{right}}$ and $\overline{M_h^{left}}$ are the average horizontal motion of pixels on the right and on the left, and k_{psi} is a proportionality factor, the helicopter must turn according to a value proportional to the difference of motions measured on both its sides to reduce this difference.

Furthermore, the high-level controller is endowed with a second reflex that allows the robot to avoid hitting a wall directly in front of it, a situation in which lateral optic flows are equal on both sides of the robot. This reflex calls upon an estimate of the time-to-contact τ, a rough approximation of which is sufficient to prevent the helicopter from crashing. If P designates a perceived point, $\overrightarrow{r}$ its projection onto the image plane and $\overrightarrow{M_r}$ the motion of $\overrightarrow{r}$, and if r and M_r refer to the norms of $\overrightarrow{r}$ and $\overrightarrow{M_r}$ in the image frame (Fig. 6), we can deduce Eq. (7) from Eq. (4), replacing h by r:

$$\frac{M_r}{r} = \frac{V_x}{X} = \frac{1}{\tau} = \eta \tag{7}$$

Therefore, we may compute η by averaging this equation over every pixel of

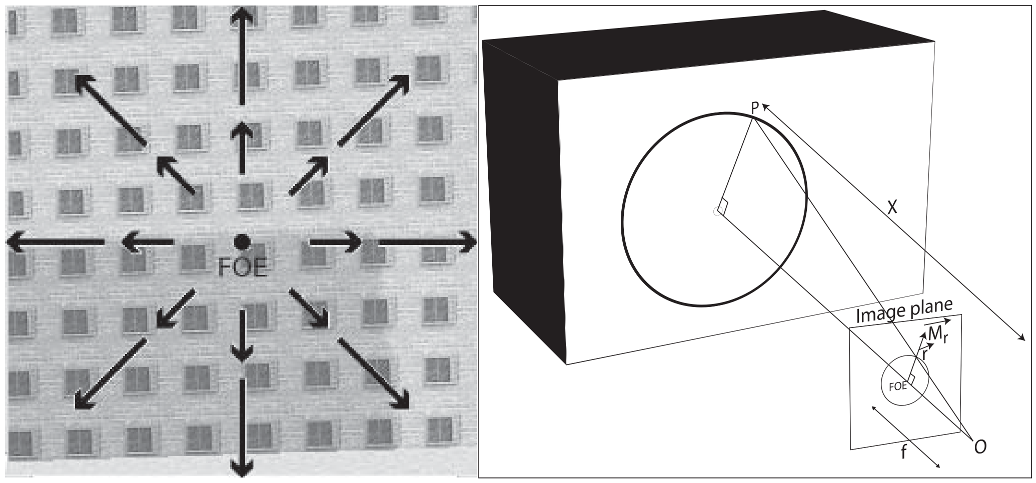

Fig. 6. Left: When the helicopter is facing an obstacle, the average horizontal optic flows are equal on both sides. However, the frontal optic flow may be used to trigger a U-turn reflex when η exceeds η_{max} (see text). Right: When the helicopter in O perceives a point P, $\overrightarrow{r}$ and $\overrightarrow{M_r}$ are respectively the projection of P on the image plane and its apparent motion.

the image (Fig. 6):

$$\eta = \overline{\left(\frac{\overrightarrow{M_r}.\overrightarrow{r}}{r^2}\right)} \tag{8}$$

where $\frac{\overrightarrow{M_r}.\overrightarrow{r}}{r}$ measures the consistence of the detected motion $\overrightarrow{M_r}$ with the theoretical radial expansion of the FOE.

The η value, the inverse of τ, informs about the risks of a frontal collision. If η exceeds η_{max}, then such a collision is likely to occur, and the high-level controller triggers a U-turn, V_x^{target} being set to zero to reduce this risk during the manoeuvre. As the helicopter reminds blind and does not make any decision meanwhile, the controller used in this work implements a *subsumption architecture* [5] according to which the U-turn reflex has a higher priority level than the balance strategy. Moreover, as the optic flow algorithm can detect motions in only a limited range (from $\frac{p}{D}$ to p pixels), care must be taken to avoid pixel motions going beyond these bounds. As the horizontal pixel motion is proportional to the ratio $\frac{V_x}{d}$, the longitudinal speed V_x must be adapted to the average distance to obstacles. The more cluttered the environment, the slower the robot must fly, and vice versa. Furthermore, this behavior will contribute to increasing the survival capacity of the helicopter because it will slow it down in dangerous zones. For this purpose, we modulate V_x^{target} according to:

$$\Delta\left(V_x^{target}\right) = k_{V_x}\left(\eta^{target} - \eta\right) \tag{9}$$

where k_{V_x} is a proportional factor and η^{target} is the value we want η to stay close to.

The strategy just described works if the helicopter's displacement is a forward translation. Unfortunately, the rotational component of the optic flow does not depend on distance and, therefore, corrupts the measurement of the translational component. This is why, in a previous work [29], we draw inspiration from the *zig-zag* flight of the housefly and implemented a helicopter controller that alternated strict straightforward flight periods - during which valid control decisions could be made - and tight-turn periods - during which no decisions were made. Although this controller led to acceptable results, it has been improved here to relax the constraint of straightforward flight periods.

3.3 Compensation for the rotational component of optic flow

Actually, if the housefly seems to be shortening its turning periods as much as possible, it nevertheless does not give up any control ability and uses the inertial information provided by its halteres to compensate for the rotational component of the optic flow. To implement such a solution, we used the balance strategy defined by Eq. (6), according to which the helicopter was free to move forward and skirt around obstacles, while blind periods were restricted to occasions when U-turns were triggered. As for the optic flow, it was managed according to the following rationale.

The Coriolis equation of a moving coordinate system, with translational velocity $\overrightarrow{V}_{helicopter}$ and rotational velocity $\overrightarrow{\Omega}_{helicopter}$, results in a motion $\overrightarrow{V}_p$ of a point $\overrightarrow{P}$ in the environment with respect to this coordinate system:

$$\overrightarrow{V}_p = -\overrightarrow{V}_{helicopter} - \overrightarrow{\Omega}_{helicopter} \times \overrightarrow{P} \tag{10}$$

with:

$$\overrightarrow{P} = (P_x, P_y, P_z) \tag{11}$$

$$\overrightarrow{V}_{helicopter} = (V_x, V_y, V_z) \tag{12}$$

$$\overrightarrow{\Omega}_{helicopter} = (\omega_x, \omega_y, \omega_z) \tag{13}$$

From Eq. (10), knowing $\overrightarrow{V}_p$ and $\overrightarrow{\Omega}_{helicopter}$, one may deduce the translational component $\overrightarrow{V}_{helicopter}$ that is needed for obstacle-avoidance. To this end, we use a pinhole model of the camera: the projection of a point $\overrightarrow{P}$ in the environment onto a point $\overrightarrow{r}$ on the image plane (Fig. 6) leads to:

$$\overrightarrow{r} = \frac{f}{P_x}.\overrightarrow{P} = (f, r_i, r_j) \tag{14}$$

Projecting the motion of $\overrightarrow{P}$ onto the image plan leads to the velocity $\overrightarrow{M}_r$ of the point $\overrightarrow{r}$ on the image plane:

$$\overrightarrow{M}_r = f.\begin{pmatrix} \frac{-V_y + r_i.V_x}{P_x} + \omega_y(r_i.r_j) - \omega_z(1 + r_i^2) + \omega_x.r_j \\ \frac{-V_z + r_j.V_x}{P_x} + \omega_y(1 + r_j^2) - \omega_z(r_i.r_j) - \omega_x.r_i \end{pmatrix} \tag{15}$$

Because the low-level controller prevents the helicopter from skidding aside, the lateral velocity V_y can be set to zero. We can also consider V_z as null because the helicopter flies at a constant altitude. The above equation can be accordingly simplified:

$$\overrightarrow{M}_r = f.\begin{pmatrix} \frac{r_i.V_x}{P_x} + \omega_y(r_i.r_j) - \omega_z(1 + r_i^2) + \omega_x.r_j \\ \frac{r_j.V_x}{P_x} + \omega_y(1 + r_j^2) - \omega_z(r_i.r_j) - \omega_x.r_i \end{pmatrix} \tag{16}$$

Let the following be defined:

$$\overrightarrow{M}^{trans} = f.\frac{V_x}{P_x}.\begin{pmatrix} r_i \\ r_j \end{pmatrix} \tag{17}$$

and:

$$\overrightarrow{M}^{rot} = f.\begin{pmatrix} \omega_y(r_i.r_j) - \omega_z(1 + r_i^2) + \omega_x.r_j \\ \omega_y(1 + r_j^2) - \omega_z(r_i.r_j) - \omega_x.r_i \end{pmatrix} \tag{18}$$

$\overrightarrow{M}^{trans}$ is another expression of Eq. (5), the information we need to avoid obstacles. The motion field is a simple summation:

$$\overrightarrow{M} = \overrightarrow{M}^{trans} + \overrightarrow{M}^{rot} \tag{19}$$

In our approximation of the motion field through an optic flow extraction algorithm, pixels that appear non-moving are not reliable because they may actually be moving too quickly to be detected. We consequently do not take these pixels into account. On the contrary, for each moving pixel, we compute $\overrightarrow{M}^{rot}$ and subtract it from the optic flow measurement to estimate $\overrightarrow{M}^{trans}$.

At this point, we use Eqs. 6, 8 and 9 at each time step of the simulation to secure a reactive behavior. Introducing a low-pass filter on η, $\overline{M_h^{right}}$ and $\overline{M_h^{left}}$ (see Fig. 2) makes the resulting strategy more efficient, even if it adds a delay in the control loop. We use a simple linear first-order filter:

$$A_{filtered} = \alpha . A + (1 - \alpha) . A_{filtered} \tag{20}$$

with: $\alpha = 0.075$ and A being one of the three values to be filtered.

Finally, we also use a PID module (Balance PID in Fig. 2) in the computation of $\Delta\psi^{target}$ so its derivative component dampers possible oscillations.

4 Experimental results

The controller described above has been put to work in three different urban environments. They are enclosed within 100m-wide cubes, while the main rotor of the simulated helicopter has a length of 150 cm. They are closed by detectable walls, thus making it possible for the helicopter to wander inside for a long time if no crash occurs. One hundred experiments, lasting five minutes each, were performed in each environment. The starting point of each such run was a randomly-chosen position, with no frontal obstacle nearby. The first environment includes six buildings which cover around 20 percent of the ground surface. The second environment includes 11 buildings covering around 30 percent of the ground surface. The third environment includes 18 buildings covering around 40 percent of the ground surface. In each environment, different textures were associated with the walls of each building.

We used a simulated camera with a field of view of 60 degrees, with a real-time quantized optic flow algorithm using a D value of 10 steps, a 3×3 patch P_v and a p parameter of 4 to pool pixels into blocks. A visual input flow of 25 simulated images of 256×256 pixels per second was thus generated and served to control the helicopter (Fig. 7). Finally, the noise that was added to, or subtracted from, each input value was drawn from a normal distribution, with a mean of 0 and a standard deviation of 0.1 m for V_x and V_y, 0.25 m for Z, and 1 deg for ϕ, θ and ψ. These values are expected to correspond to measurement errors of usual sensors.

Under these conditions, it appears (Tab. 1) that the helicopter never hit an obstacle in the three environments. Concerning velocity control, the average velocity of the helicopter decreased with the number of obstacles from 1.06

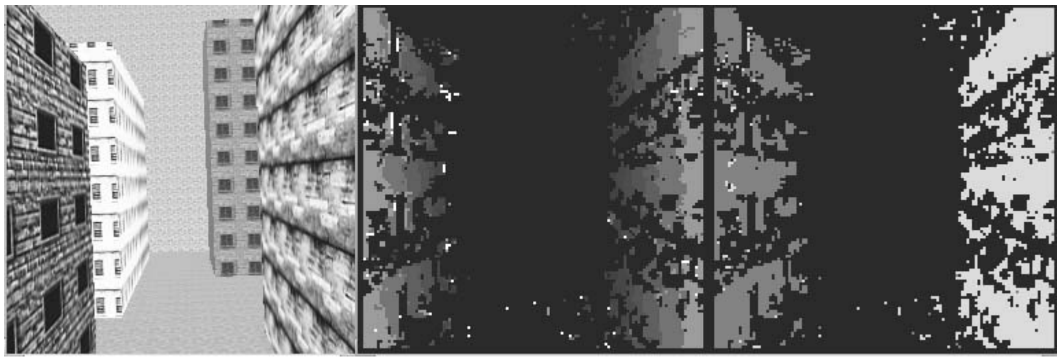

Fig. 7. The left image represents the simulated environment. The central image indicates the velocity: clearer pixels move faster. The right image shows horizontal motions detected by the optic flow algorithm: black pixels are not moving, light-grey pixels are moving towards the left, dark-grey pixels are moving towards the right.

	env. 1	env. 2	env. 3
survival rate (%)	100	100	100
average velocity (m/s)	1.0630	0.9831	0.7377
standard deviation of velocity (m/s)	0.3017	0.2504	0.2011

Table 1
Experimental results obtained in three different environments. 100 experiments, lasting 300 seconds each, were performed in each environment.

m/s in environment 1 to 0.74 m/s in environment 3. The survival rate indicates that the high-level controller succeeds in adapting the helicopter's velocity to its local environment, confining it to a range (from p to $\frac{p}{D}$) where the optic flow extraction algorithm works correctly. The standard deviation of velocity is closely linked to the standard deviation of the mean obstacle distance: in cluttered environments, this value remains relatively constant and low, whereas, in less cluttered environments, it ranges from high values - when the robot is in an open space - to low values - when the robot reaches a corridor or approaches a wall. Fig. 8 illustrates the controller's ability to slow down in cluttered environments and to speed up in open spaces. Similar information is provided by Fig. 9, which describes how the average velocity changes in each location on the map.

Likewise, Fig. 10 shows that the helicopter's trajectories remain close to the skeleton of each environment i.e., to the set of points equidistant from surrounding obstacles. Fortunately, this is particularly true in the most cluttered environment. In open spaces, the obstacles may be far away, preventing the system from detecting many moving pixels. In these cases, the resulting decision is less reliable, and the helicopter is less committed to remaining near the skeleton. However, as shown above, this behavior does not affect the survival rate.

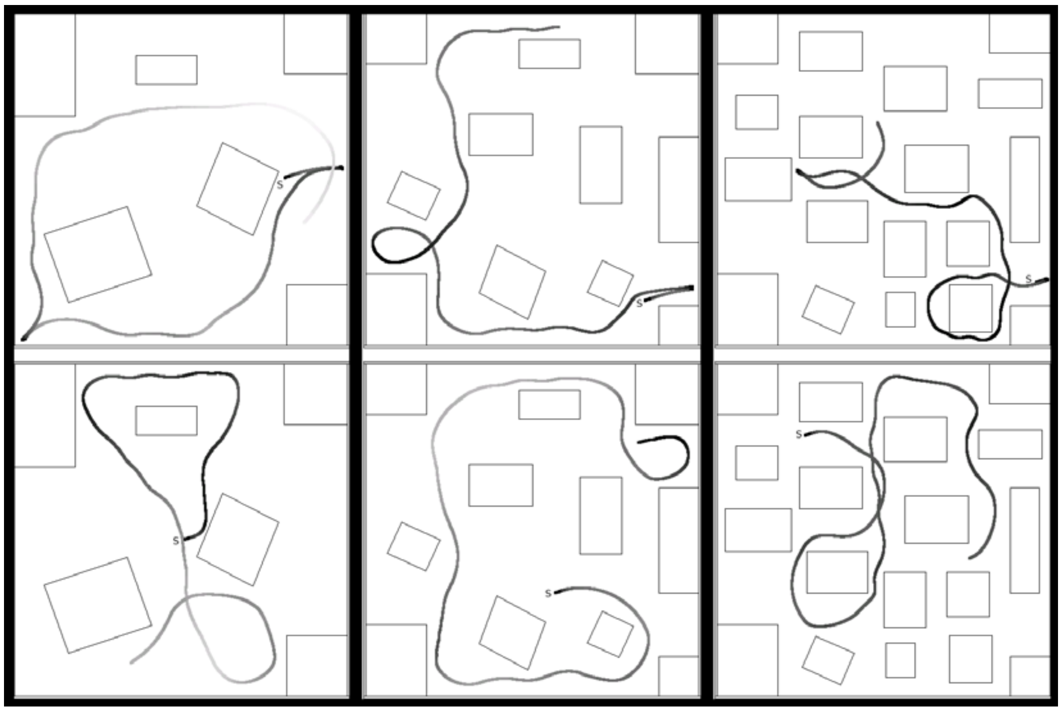

Fig. 8. Two specific trajectories among one hundred are illustrated in each environment, S being the starting position. The helicopter is able to make U-turns and to escape from dead-ends. As the grey-level of the trajectories shown is proportional to the current speed, they demonstrate that the helicopter slows down in cluttered regions and speeds up in open spaces.

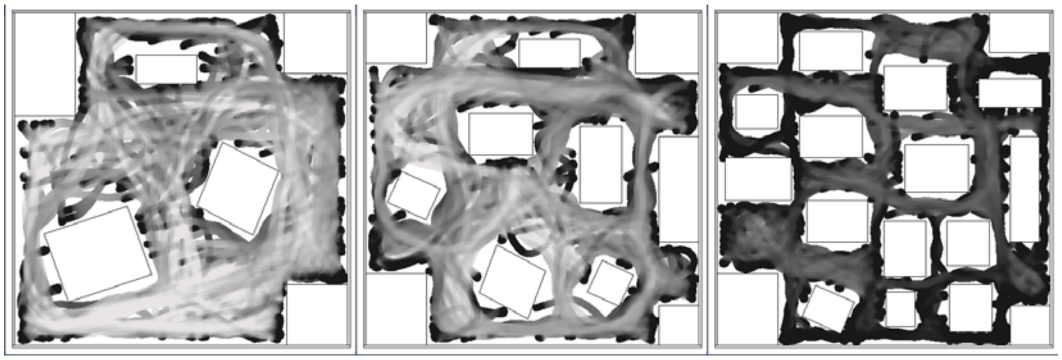

Fig. 9. Average velocity in each position, calculated over 100 runs in each environment. Greater velocities correspond to light-grey pixels.

Finally, in order to estimate the robustness of our controller with respect to sensor errors, 5 series of 20 experiments using different noise levels were performed in environment 3 - the most cluttered. To this end, the nominal standard deviations values given above were multiplied by an increasing noise factor and the corresponding survival rate and average survival time were recorded (Fig. 11). It thus appears that, with twice the nominal noise level, no crashes still occur. It is only beyond unrealistic noise levels, attaining or exceeding three times the nominal values, that crashes cannot be avoided.

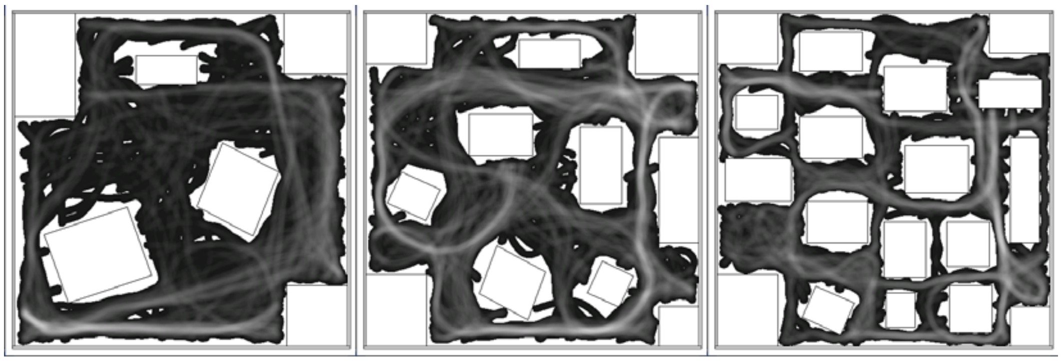

Fig. 10. Average presence of the helicopter in each position, calculated over 100 runs in each environment. Greater presence corresponds to light-grey pixels.

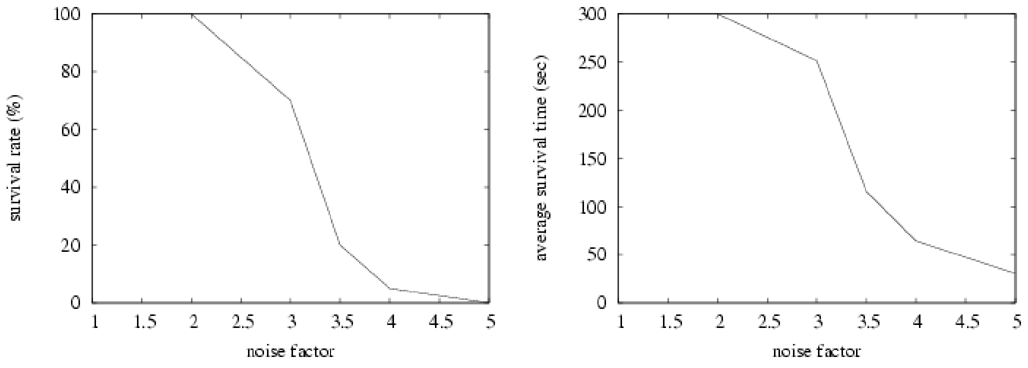

Fig. 11. Experimental results obtained in environment 3 with different noise levels. Left: survival rate. Right: average survival time. Abscissae represent the noise factor by which nominal values are multiplied.

5 Discussion

The system described herein calls upon an on-line monitoring of the optic flow and makes safe and adaptive helicopter flights possible, even in cluttered urban environments. This is partly due to the use of the real-time quantized algorithm, which calls upon simple computational principles and proves to be less time-consuming and more robust than other approaches we tried to implement. This is also due to the compensation of the rotational component of the optic flow, which allows higher speeds and smoother trajectories than the implementation of any zig-zag strategy. It turns out that the latter result is mostly due to the right tuning of the η^{target} parameter. Indeed, increasing this value too much would lead to some crashes during U-turns because, V_x^{target} being set to zero, the PID modules on V_x and ψ would not be efficient enough to prevent the helicopter from leaving its *flight envelope*[6].

[6] i.e., the state variables domain in which the aircraft must remain in order to be controllable.

In its current stage of development, the control system described here is rather simple. Being inspired by behavior-based robotics [2] and the work of Gibson [19,20], it tends to minimize the information acquisition and processing necessary to control the implemented behaviors. In particular, it does not require managing the kind of internal representations that would be necessary for cognitive mapping, self-localisation and trajectory planning, for instance [15,27]. However, with the help of an additional GPS module, it could easily be endowed with goal-directed navigation capacities, as demonstrated by the work of Duchon[11]. The solution would be to add a bias γ to Eq. 6, which would become:

$$\Delta\psi^{target} = k_{psi}.\left(\overline{M_h^{right}} - \overline{M_h^{left}}\right) + \gamma \tag{21}$$

where:

$$\gamma = \beta \times perc \times \left(1 - \frac{\Sigma}{a}\right) \tag{22}$$

Variable β designates the bearing of the goal with respect to the direction of the helicopter, and may be deduced from the GPS information. To prevent the helicopter from turning instantaneously toward the goal, only a given percentage *perc* of β is used to modify the current direction. Likewise, to avoid aiming directly at a goal that is on the other side of a wall, the value of Σ, i.e., the sum of the norms of the detected motions, is taken into account. As a is an appropriately scaled constant, the term $\left(1 - \frac{\Sigma}{a}\right)$ is necessary to balance obstacle-avoidance reflexes and goal-directed behaviors. In dangerous situations, the helicopter will be able to turn away from the goal because, when Σ increases, γ decreases.

Several other research efforts have aimed at implementing biomimetic visual controllers on simulated or real UAVs. In particular, Neumann and Bülthoff [32–34] designed a simulated UAV which was able to stabilize its attitude and avoid obstacles thanks to optic flow computation in a spherical field of view using *Elementary Motion Detectors* (EMD) that are inspired by the housefly's visual system. However, the corresponding simulation, unlike ours, does not take inertia into account, which is acceptable for very small platforms with a low Reynolds number, such as insects, but is not realistic for larger aircrafts.

Franceschini et al. [18,28,39,17,31] also apply EMDs inspired by the housefly's to control real flying engines. However, the environments they use are equipped with black and white stripes perpendicular to the direction of motion detection, thus affording high contrasts and reliable estimates of optic flow. Moreover, the control laws they designed concern one degree of freedom only, as the corresponding engines are tethered to a rotating support and can only change their altitude. Reiser and Dickinson's approach [35] is very similar and implements EMD-based mechanisms for saccade initiation and visually controlled translational velocity in a gantry fly moving in a highly-contrasted experimental setup. As of today our system has been tested in simulation only, but it deals with more degrees of freedom and more challenging environments than those just quoted.

In a general review of biomimetic visual sensing applied to flight control, Barrows et al. [4] report that they used a biomimetic visual sensing approach to control the altitude of a real helicopter. They also mention that they developed an autopilot fusing optic flow measurement and inertial information to detect nearby objects. However, too few details are provided to make a comparison with the results presented here possible. Nevertheless, this review strongly suggests that the technology advocated here has already been successfully implemented on real flying platforms.

Subsequent work will be aimed at filling the reality gap and at implementing this control system on a Concept 60 SR II engine, for which several characteristics of the simulator were specifically tailored. Moreover, great care has been taken to capitalize on simulated sensors that are easily available off the shelf. In a first stage, the controller will be tested with images acquired during real helicopter flights, as done in [22], to check the efficiency of the optic flow processing and to assess whether on-board vibration problems need to be solved. Later work will be dedicated to coping with air turbulences, which are notoriously difficult to simulate realistically when urban environments are concerned.

6 Conclusion

This work describes how a bio-inspired vision system, which monitors optic flow in real time, can be used to control the flight of a simulated autonomous helicopter in an unknown urban environment. The corresponding controller takes inertial information into account to separate the rotational and lateral components of the optic flow. This implementation generates cautious behavior, whereby the helicopter tends to stay in the middle of narrow corridors, while its forward velocity is automatically reduced when the obstacle density increases. Moreover, when heading into a frontal obstacle, the controller is able to generate a tight U-turn that ensures the UAV's survival.

When compared to related works, the simulations described here prove in some ways more realistic than other approaches where real helicopters were used, but with prepared environments and/or freedom restrictions. Moreover, it seems that no convincing reports of the successful implementation of obstacle-avoidance mechanisms on an autonomous real helicopter have yet been published. This will be the objective of our future work.

References

[1] P. Anadan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[2] R. Arkin. *Behavior-Based Robotics.* The MIT Press, 1998.

[3] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. System and experiment performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[4] G. L. Barrows, J. S. Chahl, and M. V. Srinivasan. Biomimetic visual sensing and flight control. *The Aeronautical Journal, London: The Royal Aeronautical Society*, 107(1069):159–168, 2003.

[5] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

[6] T. A. Camus. Calculating time-to-collision with real-time optical flow. *Visual Communications and Image Processing '94*, September 1994.

[7] T. A Camus. *Real-Time Optical Flow.* PhD thesis, Brown University, Departement of Computer Science, Providence, RI 02912, USA, 1994.

[8] T. A. Camus. Real-time quantized optical flow. *The Journal of Real-Time Imaging*, 3:71–86, 1997.

[9] P. Cobos and F. Monasterio. Fpga implementation of camus correlation optical flow algorithm for real time images. In *Proceedings of the Victorians Institute Conference*, pages 7–9, 2001.

[10] T. S. Collett and L. I. K. Harkness. Depth vision in animals. In D. J. Ingle, M. A. Goodale, and R. J. W. Mansfield, editors, *Analysis of Visual Behavior*, pages 111–176, Cambridge, MA, 1982. MIT Press.

[11] A. P. Duchon. Maze navigation using optical flow. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. Wilson, editors, *From Animals to Animats 4, Proceedings of the International Conference on Simulation of Adaptive Behavior*, pages 224–232, Cambridge, MA, 1996. MIT Press/Bradford Books.

[12] A. P. Duchon and W. H. Warren. Robot navigation from a gibsonian viewpoint. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 2272–2277, Piscataway, NJ, 1994. IEEE.

[13] A. P. Duchon, W. H. Warren, and L. Pack Kaelbling. Ecological robotics: Controlling behaviour with optical flow. In *Proceedings of the 17th Annual Conference of The Cognitive Science Society*, pages 164–169, Mahwah, NJ, 1995. Lawrence Erlbaum Associates.

[14] A. P. Duchon, W. H. Warren, and L. Pack Kaelbling. Ecological robotics. *Adaptive Behavior, Special Issue on Biologically Inspired Models of Spatial Navigation*, 6(3/4):473–507, 1998.

[15] D. Filliat and J.-A. Meyer. Map-based navigation in mobile robots - i. a review of localization strategies. *Journal of Cognitive Systems Research*, 4(4):243–282, 2003.

[16] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[17] N. Franceschini. From fly vision to robot vision: Re-construction as a mode of discovery. In F. Barth, J. A. C. Humphrey, and Secomb T., editors, *Sensors and Sensing in Biology and Engineering*, pages 223–236. Springer, Berlin, 2002.

[18] N. Franceschini, J. M. Pichon, and C. Blanes. From insect vision to robot vision. *Philosophical Transactions of the Royal Society of London B*, 4(4):283–294, 1992.

[19] J. J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, 1950.

[20] J. J. Gibson. *The Ecological Approach to visual Perception*. Houghton Mifflin, 1979.

[21] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–204, 1981.

[22] S. Hrabar and G. S. Sukhtame. A comparison of two camera configurations for optic-flow based navigation of a uav through urban canyons. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2004. In Press.

[23] D. N. Lee. A theory of visual control based on information about time-to-collision. *Perception*, 5:437–459, 1976.

[24] D. N. Lee. The optic flow field: The foundation of vision. *Philosophical Transactions of the Royal Society of London B*, 290:169–179, 1980.

[25] D. N. Lee and P. E. Reddish. Plummeting gannets: A paradigm of ecological optics. *Nature*, 293:293–294, 1981.

[26] B. D. Lucas and T. Kanade. An iterative image-registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*, pages 121–130, 1981.

[27] J.-A. Meyer and D. Filliat. Map-based navigation in mobile robots - ii. a review of map-learning and path-planning strategies. *Journal of Cognitive Systems Research*, 4(4):283–317, 2003.

[28] F. Mura and N. Franceschini. Visual control of altitude and speed in a flying agent. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3, Proceedings of the third international conference on*

Simulation of Adaptive Behavior, Cambridge, MA, 1994. MIT Press/Bradford Books.

[29] L. Muratet, S. Doncieux, and J.-A. Meyer. A biomimetic reactive navigation system using the optical flow for a rotary-wing uav in urban environment. *In Proceedings of the International Session on Robotics*, March 2004.

[30] T. Netter and N. Franceschini. Neuromorphic optical flow sensing for nap-of-the-earth flight. In *Mobile Robots XIV*, volume 3838 of *SPIE*, pages 208–216, Bellingham, USA, 1999.

[31] T. Netter and N. Franceschini. A robotic aircraft that follows terrain using a neuromorphic eye. In *Proceedings of IEEE Conference on Intelligent Robots and Systems (IROS)*, pages 129–134, Lausanne, Switzerland, 2002.

[32] T. R. Neumann and H. H. Bülthoff. Biologically motivated visual control of attitude and altitude in translatory flight. In G. Baratoff and H. Neumann, editors, *In Proceedings of Artificial Intelligence: Proceedings of the 3rd Workshop Dynamische Perzeption*, volume PAI 9, pages 135–140, Berlin, 2000. Infix-Verlag.

[33] T. R. Neumann and H. H. Bülthoff. Insect inspired visual control of translatory flight. In *Advances in Artificial Life - Proceedings of the 6th European Conference on Artificial Life ECAL 2001*, volume LNCS/LNAI 2159, pages 627–636. Springer-Verlag, 2001.

[34] T. R. Neumann and H. H. Bülthoff. Behavior-oriented vision for biomimetic flight control. In *Proceedings of the EPSRC/BBSRC International Workshop on Biologically Inspired Robotics: The Legacy of W. Grey Walter, 14-16 August 2002, HP Labs Bristol, UK*, pages 196–203, 2002.

[35] M. B. Reiser and M. H. Dickinson. A test bed for insect-inspired robotic control. *Philosophical Transactions of the Royal Society of London A*, 361:2267–2285, 2003.

[36] M. V. Srinivasan, J. S. Chahl, K. Weber, S. Venkatesh, M. G. Nagle, and S. W. Zhang. Robot navigation inspired by principles of insect vision. *Field and Service Robotics*, pages 12–16, 1998.

[37] M. V. Srinivasan, Lehrer S. W., W. M. Kirchner, M., and S. W. Zhang. Range perception through apparent image speed in freely flying honeybees. *Visual Neuroscience*, 6:519–535, 1991.

[38] M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett. Honeybee nagigation en route to the goal: Visual flight control and odometry. *The Journal of Experimental Biology*, 199:237–244, 1996.

[39] S. Viollet and N. Franceschini. Super-accurate visual control of an aerial minirobot. *Autonomous minirobots for Research and Edutainment*, pages 215–224, 2001.