

# Design and Control of a Miniature Quadrotor

**Book Chapter****Author(s):**

Bouabdallah, Samir; Siegwart, Roland

**Publication date:**

2007

**Permanent link:**

<https://doi.org/10.3929/ethz-a-010118239>

**Rights / license:**

[In Copyright - Non-Commercial Use Permitted](#)

**Originally published in:**

Intelligent Systems, Control and Automation: Science and Engineering 33, [https://doi.org/10.1007/978-1-4020-6114-1\\_6](https://doi.org/10.1007/978-1-4020-6114-1_6)

# Design and Control of a Miniature Quadrotor

Samir Bouabdallah and Roland Siegwart

Autonomous Systems Lab, ETH Zurich, Switzerland



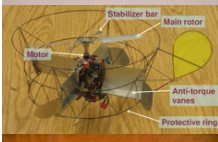
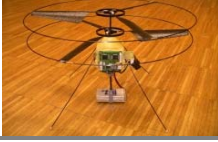
## Introduction

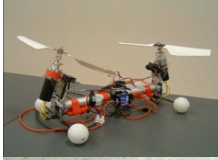


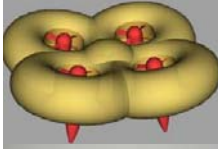


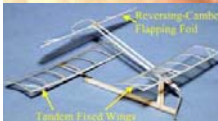
In the last ten years Miniature Aerial Vehicles (MAV) have gained a strong interest. The recent advances in low-power processors, miniature sensors and control theory are opening new horizons in terms of miniaturization and field of application. Miniature Flying Robots (MFR) show all their advantages in complex or cluttered environments. This is the scenario of office buildings and commercial centers that are among many other areas for aerial surveillance. MFR can also serve in search-and-rescue missions after earthquakes, explosions, etc. These are most of the time dangerous scenarios and require important task-forces including fire-fighters and Samaritans. An aerial robot capable of flying in narrow space and collapsed buildings could quickly and systematically search victims of accidents or natural disasters without risking human lives. When flying in such conditions, it is essential to have a vehicle that can easily fit through small openings and maneuver around pillars and destructed wall structure. MFR could, after localization, provide coordinates of people to guide rescue forces in their task. In such hazardous environment, wireless communication is difficult and very often impossible because of natural obstacles. An aerial relay would be possible using several aerial vehicles equipped with transceivers, ensuring a constant communication. The potential capabilities of these systems and the challenges behind are attracting the scientific and the industrial communities.

Paper (Hoffmann et al. 2004) outlined the development of a miniature autonomous flight control system and the creation of a multi-vehicle platform for experimentation and validation of multi-agent control algorithms.

Document (Kroo et al. 2000) presents several results in centimeter-scale quadrotor design and analysis. One recent result from (Wang et al. 2006) is a 13.6 cm micro-helicopter able to hover 3 minutes. The Swiss Federal Institute of Technology is also participating with several projects to this scientific endeavor (AERO 2006). At the Autonomous Systems Lab we are convinced that the emergence of fully autonomous MFRs will be the result of a system-level optimization and a simultaneous effort on design and control. Our approach is mainly to design vehicles with optimized mechanics and apply to them innovative control techniques. The idea is to miniaturize the robot in every redesign step in order to take benefit from the latest technological advancements. The objective of ASL-MFR project is to optimally design and control aerial systems for navigation in cluttered environments. A quadrotor (OS4) and a coaxial (CoaX) Vertical Take-Off and Landing (VTOL) systems are particularly considered because of their challenging control problem and their broad field of application. Table 6.1 lists different configurations commonly used in MAV research and industry.

**Table 6.1.** Common UAV-MAV configurations

Configuration	Picture	Advantages	Drawbacks
Fixed-wing (AeroVironment)		- Simple mechanics - Silent operation	- No hovering
Single (A. V de Rostyne)		- Good controllability and maneuverability	- Complex mechanics - Large rotor - Long tail boom
Axial rotor (Maryland Univ.)		- Compactness - Simple mechanics	- Complex control - Weak maneuverability
Coaxial rotors (ETHZ)		- Compactness - Simple mechanics	- Complex aerodynamics

Tandem rotors (Heudiasyc)		- Good controllability and maneuverability - No aerodynamics interference	- Complex mechanics - Large size
Quadrotors (ETHZ)		- Good maneuverability - Simple mechanics - Increased payload	- High energy consumption - Large size
Blimp (EPFL)		- Low power consumption - Auto-lift	- Large size - Weak maneuverability
Hybrid (MIT)		- Good maneuverability - Good survivability	- Large size - Complex design
Bird-like (Caltech)		- Good maneuverability - Low power consumption	- Complex mechanics - Complex control
Insect-like (UC Berkeley)		- Good maneuverability - Compactness	- Complex mechanics - Complex control
Fish-like (US Naval Lab)		- Multimode mobility - Efficient aerodynamics	- Complex control - Weak maneuverability

In fact, in comparison to other flying principles, VTOL systems have specific characteristics which allow the execution of applications that would be difficult or impossible with other concepts. Table 6.2 gives a short and not exhaustive comparison between different VTOL concepts. This is an adaptation of the larger comparison in (Datta et al. 2000). One can see in this table that the quadrotor and the coaxial helicopter are among the best configurations if used as MFR.

**Table 6.2.** VTOL concepts comparison (1=Bad, 4=Very good)

	A	B	C	D	E	F	G	H
Power cost	2	2	2	2	1	4	3	3
Control cost	1	1	4	2	3	3	2	1
Payload/volume	2	2	4	3	3	1	2	1
Maneuverability	4	2	2	3	3	1	3	3
Mechanics simplicity	1	3	3	1	4	4	1	1
Aerodynamics complexity	1	1	1	1	4	3	1	1
Low speed flight	4	3	4	3	4	4	2	2
High speed flight	2	4	1	2	3	1	3	3
Miniaturization	2	3	4	2	3	1	2	4
Survivability	1	3	3	1	1	3	2	3
Stationary flight	4	4	4	4	4	3	1	2
<b>Total</b>	<b>24</b>	<b>28</b>	<b>32</b>	<b>24</b>	<b>33</b>	<b>28</b>	<b>22</b>	<b>24</b>

A=Single rotor, B=Axial rotor, C=Coaxial rotors, D=Tandem rotors, E=Quadrotor, F=Blimp, G=Bird-like, H=Insect-like.

This chapter is organized in six sections. Modelling for simulation is presented in the second section. The third section presents the OS4 simulator. The fourth section focuses on design methodology and its application to OS4 design. Section five is dedicated to simulation and control. Finally, section six is devoted to a conclusion and review of future work.

## Modelling for Simulation

The simulation model of OS4 was developed through several successive steps as presented in (Bouabdallah et al. 2004, 2005). The major improvements of this version are the inclusion of hub forces ( $H$ ), rolling moments ( $R_m$ ) and variable aerodynamical coefficients. This makes the model more realistic particularly in forward flight. With the preliminary versions of the model it was often necessary to slightly adjust the control parameters for successful experiments. Most of these experiments were performed on the previous OS4 test-bench (Bouabdallah and Siegwart 2005). This chapter presents the model used for the last version of the OS4 simulator with which the Integral Backstepping (IB) controller was developed. This time, the simulated control parameters were directly used on the real helicopter for successful autonomous flights with the new OS4.

The dynamics of a rigid body under external forces applied to the center of mass and expressed in the body fixed frame as shown in (Murray et al. 1994) are in Newton-Euler formalism:

$$\begin{bmatrix} mI_{3 \times 3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times mV \\ \omega \times I\omega \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \quad (6.1)$$

Where,  $I \in \mathfrak{R}^{(3 \times 3)}$  is the inertia matrix,  $V$  is the body linear speed and  $\omega$  is the body angular speed.  $F$  and  $\tau$  are respectively the body force and torque, while  $m$  is the mass of the system. Let's consider earth fixed frame  $E$  and body fixed frame  $B$  as seen in Fig. 6.1. Using Euler angles parameterization, the airframe orientation in space is given by a rotation  $R$  from  $B$  to  $E$ , where  $R$  in  $SO3$  is the rotation matrix. The frame system is slightly different comparing to previous versions in order to conform to the  $N, E, D$  (North, East, Down) standard, following by the way the coordinate system of our inertial sensor.

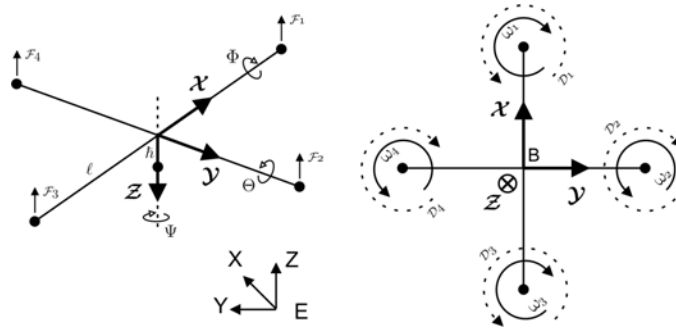


Fig. 6.1. OS4 coordinates system

## Aerodynamic Forces and Moments

The aerodynamic forces and moments are derived using a combination of momentum and blade element theory (Leishman 2005). This is based on the work of Gary Fay during Mesicopter project (Fay 2001). For an easier reading of the equations below, we recall some symbols:

Air density	$\rho$	Rotor radius	$R$
Solidity ratio	$\sigma$	Rotor speed	$\Omega$
Lift slope	$a$	Rotor area	$A$
Rotor advance ratio	$\mu$	Pitch of incidence	$\theta_0$

Inflow ratio	$\lambda$	Twist pitch	$\theta_{tw}$
Induced velocity	$v$	Average drag coefficient	$\bar{C}_d$

### **Thrust Force**

This force is the resultant of the vertical forces acting on all the blade elements of one propeller.

$$\begin{cases} T = C_T \rho A (\Omega R_{rad})^2 \\ \frac{C_T}{\sigma a} = \left(\frac{1}{6} + \frac{1}{4} \mu^2\right) \theta_0 - (1 + \mu^2) \frac{\theta_{tw}}{8} - \frac{1}{4} \lambda \end{cases} \quad (6.2)$$

### **Hub Force**

The hub force is the resultant of the horizontal forces acting on all the blade elements.

$$\begin{cases} H = C_H \rho A (\Omega R_{rad})^2 \\ \frac{C_H}{\sigma a} = \frac{1}{4a} \mu \bar{C}_d + \frac{1}{4} \lambda \mu \left(\theta_0 - \frac{\theta_{tw}}{2}\right) \end{cases} \quad (6.3)$$

### **Drag Moment**

This moment about the rotor shaft is caused by the aerodynamic forces acting on the blade elements. The horizontal forces acting on the rotor multiplied by the moment arm and integrated over the rotor. Drag moment is important as it determines the power required to spin the rotor.

$$\begin{cases} Q = C_Q \rho A (\Omega R_{rad})^2 R_{rad} \\ \frac{C_Q}{\sigma a} = \frac{1}{8a} (1 + \mu^2) \bar{C}_d + \lambda \left(\frac{1}{6} \theta_0 - \frac{1}{8} \theta_{tw} - \frac{1}{4} \lambda\right) \end{cases} \quad (6.4)$$

### **Rolling Moment**

The rolling moment of a propeller exists in forward flight when the advancing blade is producing more lift than the retreating one. It is the integration over the entire rotor of the lift of each section acting at a given ra-

dius. This is not to be confused with the overall rolling moment which is caused by a number of other effects.

$$\begin{cases} R_m = C_{R_m} \rho A (\Omega R_{rad})^2 R_{rad} \\ \frac{C_{R_m}}{\sigma a} = -\mu \left( \frac{1}{6} \theta_0 - \frac{1}{8} \theta_{tw} - \frac{1}{8} \lambda \right) \end{cases} \quad (6.5)$$

### Ground Effect

Helicopters operating near the ground experience thrust augmentation due to better rotor efficiency. It is related to a reduction of the induced airflow velocity. This is called Ground Effect. In the literature one can find different approaches to deal with this effect, for instance by using adaptive techniques (Guenard et al. 2006). However, the principal need in this project is to find a model of this effect for OS4 to improve the autonomous take-off and landing controllers. The goal is to obtain a simple model capturing mainly the variation of the induced inflow velocity. Cheeseman (Cheeseman 1957) uses the images method (Leishman 2002) to state that at constant power,  $T_{OGE} v_{i,OGE} = T_{IGE} v_{i,IGE}$ . The velocity induced at the rotor center by its image is  $\delta v_i = A v_i / 16 \pi z^2$ . Cheeseman obtained the simple relation Eq. (6.6) by assuming that  $v$  and  $\delta v_i$  are constant over the disk which allows  $v_{i,IGE} = v_i - \delta v_i$ ,  $z$  is the altitude.

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1 - \frac{R_{rad}^2}{16z^2}} \quad (6.6)$$

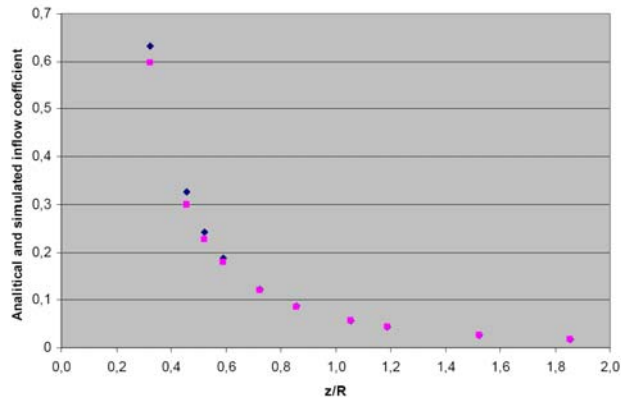
Another simple way to proceed is to consider that the inflow ratio In Ground Effect (IGE) is  $\lambda_{IGE} = (v_{i,OGE} - \delta v_i - \dot{z}) / \Omega R_{rad}$ , where the variation of the induced velocity is  $\delta v_i = v_i / (4z / R_{rad})^2$ . We can then rewrite the thrust coefficient Eq. (6.2) IGE as follows:

$$\begin{cases} T_{IGE} = C_T^{IGE} \rho A (\Omega R_{rad})^2 \\ \frac{C_T^{IGE}}{\sigma a} = \frac{C_T^{OGE}}{\sigma a} + \frac{\delta v_i}{4\Omega R_{rad}} \end{cases} \quad (6.7)$$

Then we compared the variation of the inflow velocity in and out of ground effect using OS4 simulator. In Fig. 6.2 we plot the ratio  $(\delta v_i / v_i)$  obtained by simulation and by analytical derivation. The influence is per-



ceptible for  $z/R_{rad} \approx 2$  but becomes important near  $z/R_{rad} \leq 1$ . It seems then that in the case of a quadrotor the ground effect influence is already present at one rotor diameter and becomes important at one rotor radius.



**Fig.6.2.** Simulation: Ground effect influence on the inflow velocity

In order to empirically verify this assumption, we conducted a simple experiment which proved that a quadrotor deprived of altitude control can hover at a constant altitude at nearly one rotor diameter from the ground. It is clear that this result is only an indication of validity and does not constitute a formal proof.

### Moments and Forces

Quadrotor motion is obviously caused by a series of forces and moments coming from different effects. We consider the following ones:

Roll angle	$\phi$	Vertical distance prop./CoG	$h$
Pitch angle	$\theta$	Horizontal dist. prop./CoG	$l$
Yaw angle	$\psi$	Rotor inertia	$J_r$

### **Rolling Moments**

Body gyro effect:	$\dot{\theta} \dot{\psi} (I_{yy} - I_{zz})$
Propeller gyro effect:	$J_r \dot{\theta} \dot{\Omega}_r$
Roll actuators action:	$l(-T_2 + T_4)$
Hub moment due to sideward flight:	$h \left( \sum_{i=1}^4 H_{yi} \right)$
Rolling moment due to forward flight:	$(-1)^{i+1} \sum_{i=1}^4 R_{mxi}$

### **Pitching Moments**

Body gyro effect:	$\dot{\theta} \dot{\psi} (I_{zz} - I_{xx})$
Propeller gyro effect:	$J_r \dot{\theta} \dot{\Omega}_r$
Roll actuators action:	$l(T_1 - T_3)$
Hub moment due to forward flight:	$h \left( \sum_{i=1}^4 H_{xi} \right)$
Rolling moment due to sideward flight:	$(-1)^i \sum_{i=1}^4 R_{myi}$

### **Yawing Moments**

Body gyro effect:	$\dot{\theta} \dot{\phi} (I_{xx} - I_{yy})$
Inertial counter-torque:	$J_r \dot{\Omega}_r$
Counter-torque unbalance:	$(-1)^i \sum_{i=1}^4 Q_i$
Hub force unbalance in forward flight:	$l(H_{x2} - H_{x4})$
Hub force unbalance in sideward flight:	$l(-H_{y1} + H_{y3})$

### ***Forces along z axis***

Actuators action:  $\cos \psi \cos \phi \left( \sum_{i=1}^4 T_i \right)$

Weight:  $mg$

### ***Forces along x axis***

Actuators action:  $(\sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi) \left( \sum_{i=1}^4 T_i \right)$

Hub force in x axis:  $\left( -\sum_{i=1}^4 H_{xi} \right)$

Friction:  $\frac{1}{2} C_x A_c \rho \dot{x} |\dot{x}|$

### ***Forces along y axis***

Actuators action:  $(-\cos \psi \sin \phi + \sin \psi \sin \theta \cos \phi) \left( \sum_{i=1}^4 T_i \right)$

Hub force in y axis:  $\left( -\sum_{i=1}^4 H_{yi} \right)$

Friction:  $\frac{1}{2} C_y A_c \rho \dot{y} |\dot{y}|$

### **Equation of Motion**

The equations of motion are derived from Eq. (6.1) and all the forces and moments listed in the previous paragraph.

$$\left\{ \begin{array}{l} \ddot{\phi} = \sum \tau_x / I_{xx} \\ \ddot{\theta} = \sum \tau_y / I_{yy} \\ \ddot{\psi} = \sum \tau_z / I_{zz} \\ \ddot{z} = g - \sum F_z / m \\ \ddot{x} = \sum F_x / m \\ \ddot{y} = \sum F_y / m \end{array} \right. \quad (6.8)$$

### Rotor Dynamics

OS4 is equipped with four fixed-pitch rotors (no swash plate), each one includes a BLDC motor, a one-stage gearbox and a propeller. The entire rotor dynamics were identified and validated using the Matlab Identification Toolbox. A first-order transfer function is reasonable to reproduce the dynamics between the propeller's speed set-point and its true speed.

$$G(s) = \frac{0.936}{0.178s + 1} \quad (6.9)$$

It is worthwhile to note the non-unity gain in Eq. (6.9) this is visible in Fig. 6.3, which superimposes the model output (red) and the sensor data (blue) to a step input (green). In fact, sensorless BLDC<sup>1</sup> motors require a minimum speed to run thus, the set-point does not start from zero.

---

<sup>1</sup> Brush-Less Direct Current

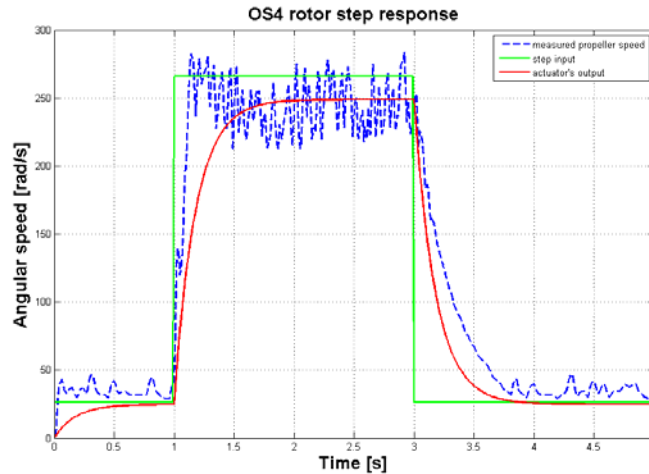


Fig.6.3. Rotor and model step response, measured at propeller shaft

## OS4 Simulator

OS4 simulator development followed the successive improvements made to the dynamic model, the control scheme and the robot hardware. The last version includes identified actuator's dynamics, aerodynamics block, obstacle avoidance controller (OAC) and a high level planner for way-points definition. Each block is described by one or several Matlab files and can be easily incorporated into other simulators. The simulation starts with the initial state taken from the dedicated block "initial conditions" (see Fig. 6.4). After that, the set of data is degraded with delay and white noise and then filtered. It is then used in the control block and the inputs are sent to the motor dynamics block. The estimated rotors' speed feeds the aerodynamics block, which outputs the forces and moments of each propeller. This is sent to the system dynamics block, along with the state and the actual rotors' speed to process the new state.

The block "control" is in fact dedicated to attitude, altitude and position controllers as schematized in Fig. 6.5. Each control loop is simulated at the sampling-time of its respective sensor.

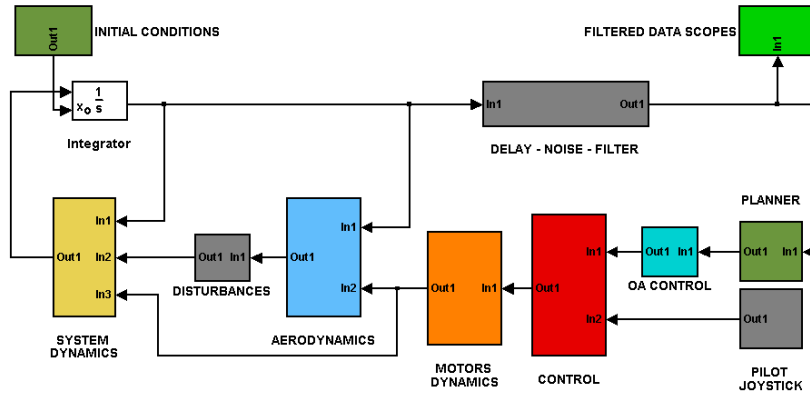


Fig.6.4. OS4 simulator block diagram

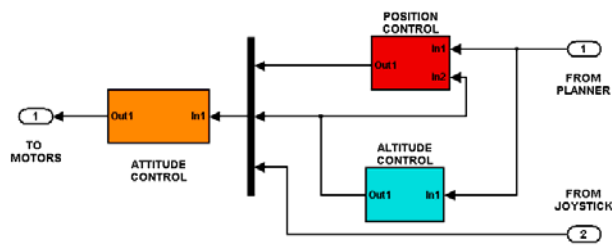


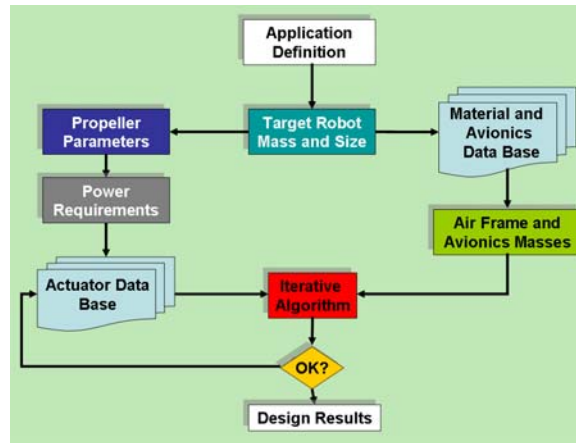
Fig.6.5. Control block in OS4 simulator

## Design

The interdependency of components during the design phase makes the choice of each one strongly conditioned by the choice of all the others and vice-versa. Starting such a design and taking a decision concerning all design variables requires following an appropriate methodology. OS4 was designed following a practical method we developed to handle the design problematic of a small scale rotorcraft. The method combines models and databases of components and produces the best selection. Moreover, it provides the required battery mass to use in order to comply with the total mass constraint.

## The General Method

The design process starts by defining three design constraints for the helicopter: Maximum mass  $m_{\max}$ , maximum span  $s_{\max}$  and target thrust/weight ratio  $T_w$ . This gives a good idea about the propeller diameter to use  $d_{prop}$ . In practice, the propeller span defines the overall span of the helicopter. Using the propeller diameter  $d_{prop}$ , one can estimate the characteristics of the propeller in term of thrust, drag and power for a range of angular speeds. Paper (Nicoud et al. 2002) proposes for this purpose, the models  $T \propto \Omega^2 L^4$ ,  $D \propto \Omega^2 L^5$  and  $P \propto \Omega^3 L^5$ , where  $L$  is a reference dimension, e.g. the center of the blade. So, the mass  $m_{\max}$ , the drag moment  $D$  and the thrust/weight ratio  $T_w$  are enough to fully define the motor power requirements. This allows the algorithm to select from the database a list of candidate actuators which offer the required power. Then, a rough estimation of the mass of the airframe  $m_{af}$  and avionics  $m_{av}$  is necessary to have a first estimation of the total mass without battery (see Fig. 6.6). So, the maximum mass is  $m_{\max} = m_{\max} = m_{af} + m_{av} + m_{pg} + m_{bat}$ , where  $m_{pg}$  is the mass of the propulsion group (propeller, gearbox, motor) and  $m_{bat}$  is the mass of the battery. The iterative algorithm will find  $m_{pg}$  and  $m_{bat}$  as described hereafter.



**Fig.6.6.** The design method flowchart. The user has to define a target mass and size of his system in addition to airframe and avionics mass

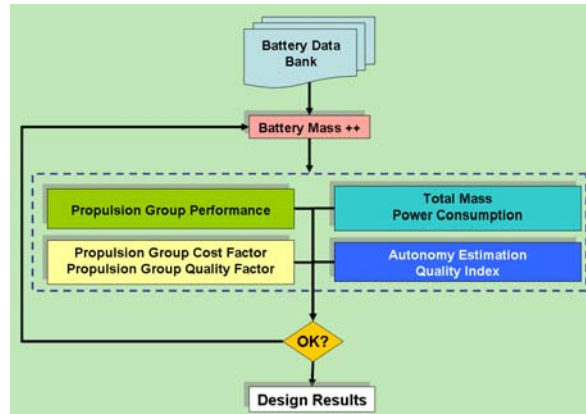
### **The Iterative Algorithm**

This algorithm starts by picking up one candidate actuator from the database, its mass is  $m_{pg}(i)$ . Then, an initial value  $m_{bat}(j_0)$  is given to  $m_{bat}$  variable. So far, we have all the variables to determine  $m_{max}(i, j) = m_{af} + m_{av} + m_{pg}(i) + m_{bat}(j)$ . The temporary total mass of the helicopter  $m_{max}(i, j)$  is used to estimate various variables (see Fig. 6.7) at two operational points: At hover and at maximum thrust. For each candidate actuator, the variable  $m_{bat}(j)$  is incremented until  $m_{max}$  is reached. This process makes it possible to estimate, at hover and at maximum thrust, for each candidate actuator, and at each increment of  $m_{bat}(j)$  the following variables:

- Total mass:  $m_{bat}$
- Total power consumption:  $P_{tot}$
- Propulsion group efficiency:  $\eta_{gb}$
- Propulsion group cost factor:  $C = P_{el} / (T - m_{pg})$
- Propulsion group quality factor:  $Q = B_w T_w / \Omega C$   
( $B_w$  : PG bandwidth,  $T_w$  : thrust/weight ratio)
- Operational time (autonomy):  $Au = m_{bat} C_{bat} / P_{el}$
- Design quality index:  $Q_{in} = Au / P_{tot}$

Propulsion group cost factor  $C$  describes the cost in power of the lifted mass. Propulsion group quality factor  $Q$  describes the quality of this mass lifting (see Table 6.4.). The propulsion group quality factor is necessary to take into account the notions of actuator bandwidth and thrust to weight ratio. On the other hand, the design quality index constraints the operational time with regard to the total power consumption.

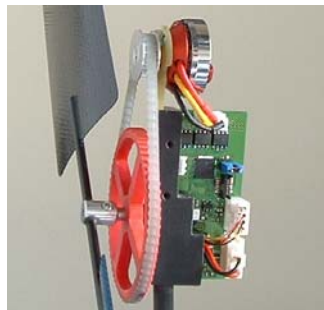




**Fig.6.7.** The iterative algorithm flowchart

### OS4 Quadrotor

The method presented before was used to design OS4 starting from two main constraints: 500 g maximum mass and 800 mm maximum span. A propeller of 300 mm in diameter was selected which respects the span constraint. The main design variables of a propulsion group are listed in Table 6.3. They were used in the models of Table 6.4.



**Fig.6.8.** OS4 propulsion group. The module is interfaced through I<sup>2</sup>C bus and has a local PI speed controller

**Table 6.3.** OS4 propulsion group design variables

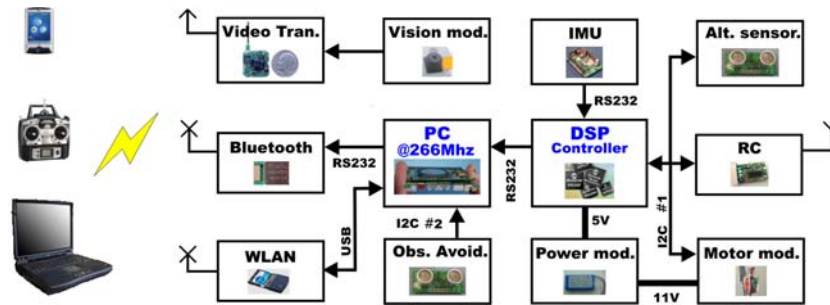
propeller		OS4	unit
mass	$m_p$	5.2	g
thrust coeff.	$b$	3.13e-5	N s <sup>2</sup>
drag coeff.	$d$	7.5e-7	Nm s <sup>2</sup>
inertia	$J_r$	6e-5	kg.m <sup>2</sup>
gearbox		OS4	unit
efficiency	$\eta$	90	%
mass	$m_{gb}$	7	g
max. torque		0.15	Nm
max. speed		1000	rad/s
red. ratio	$r$	4:1	
motor		OS4	unit
effi. at hover	$\eta_m$	64	%
mass	$m_m$	12	g
max. power	$P_{el}$	35	W
internal res.	$R_{mot}$	0.6	$\Omega$
inertia	$J_m$	4e-7	kgm <sup>2</sup>
torque cst	$k$	5.2	mNm/A

**Table 6.4.** Models of the propulsion group components

component	model
propeller	$(b, d)\Omega^2 = (T, D)$
gearbox	$P_{in}\eta = P_{out}$
DC motor	$-\frac{k^2}{R_{mot}}\omega - D + \frac{k}{R_{mot}}u = J \frac{d\omega}{dt}$
PG cost	$P_{el}/(T - m_{pg}) = C$
PG quality	$B_w T_w / \Omega C = Q$

PG=Propulsion Group (motor+geabox+propeller)

Finally, the choice of the propulsion group is done by the iterative algorithm which produces a classification based on the cost and quality factors. It provides also the battery mass to use:  $m_{bat}=230$  g (11 V, 3.3 Ah) of Lithium-Polymer. On the rotor side, the tests revealed that a gearbox is necessary. In fact, a direct-drive propulsion group would allow only a thrust to weight ratio of  $T_w=0.75$ , which is obviously not enough to lift the robot. The selected motor is a brushless DC motor (12 g, 35 W) with a high power to weight ratio, which justifies the choice even including its control electronics. A 6 g I<sup>2</sup>C controller was specially designed for the sensorless outrunner LRK195.03 motor as shown in Fig. 6.8. Obviously, BLDC motors offer high life-time and low electromagnetic noise. The ready to plug propulsion group weighs 40 g and lifts more than 260 g.



**Fig.6.9.** OS4 block diagram. A DSP processor handles attitude and altitude control. Then, a miniature PC (x-board) handles obstacles avoidance control and communication tasks. The robot communicates through a wifi interface and accepts standard remote control signals

Fig. 6.9. represents OS4's block diagram. Embedding the controller for our application is definitely advisable as it avoids all the delays and the discontinuities in wireless connections. A miniature computer module, based on Geode 1200 processor running at 266MHz with 128M of RAM and flash memory was developed. The computer module is x86 compatible and offers all standard PC interfaces. The whole computer is 44 g in mass, 56 mm by 71 mm in size and runs Linux.

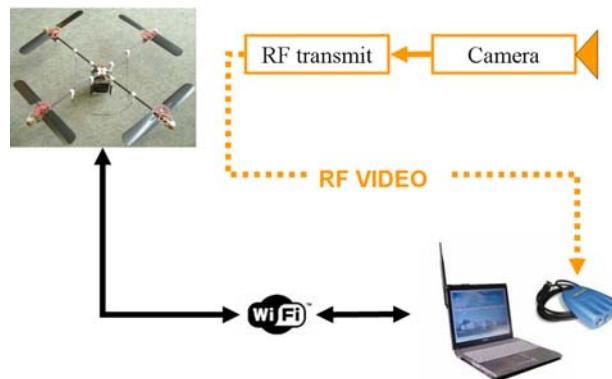


**Fig.6.10.** The x-board based, 40 g and 56x71 mm computer module

The controller includes an MCU for interfacing Bluetooth with the computer module. The same chip is used to decode the PPM signal picked up from a 1.6 g, 5 channels commercially available RC receiver. This decoding on our MCU, makes it possible to interface the RC receiver to I<sup>2</sup>C bus, and at the same time detect any anomaly in the channels. It is also possible to control the helicopter using a standard remote control.

### ***Position Sensor***

OS4's position sensor is based on an on-board down-looking CCD camera and a simple pattern on the ground. The camera provides a motion-blur free image of 320x240 at up to 25 fps. The algorithm detects the pattern, estimates the pose and provides the camera position (x,y) and heading angle ( $\psi$ ). The image is primarily sent to an off-board computer for processing and then the position data is sent-back to the helicopter for control purpose as shown in Fig. 6.11.

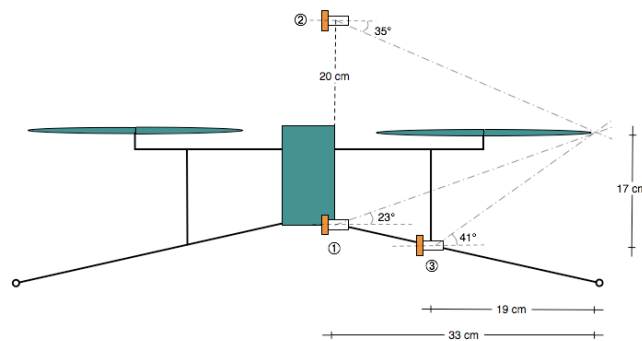


**Fig.6.11.** Position sensing setup on OS4

Several possibilities were considered for pattern detection. The first method tested is the detection of five red dots on a A4 paper. This method suffers from sensitivity to lighting conditions. For the second test, we considered four LEDs with different colors on an A4 board. However, it was hard to tune LEDs intensity for the overall working volume. Finally, we used a red A4 paper with a white spot shifted from the pattern center. This time the pattern was robustly detected. We use for that Canny edge detector and Douglas-Peucker algorithms already implemented in OpenCV. In addition, we run a least-square based linear regression to refine the detection. Pose estimation is then performed using PnP algorithm (DeMenthon et al. 2005). The sensor algorithm is afterwards enhanced with a management of different situations where the pattern is not or partially detected. All the processing takes about 7ms. Image capture takes 1ms with a PCI acquisition card and almost 20ms with a USB 1.1 device on a Pentium 4, 2.4GHz. Anyway, the algorithm is limited to 25Hz by the camera frame rate (25 fps). The errors obtained in x and y position sensing are about 2 cm at 0.5 m/s. The error on the yaw is about 3° at 180°/s.

### ***Obstacle Detection Setup***

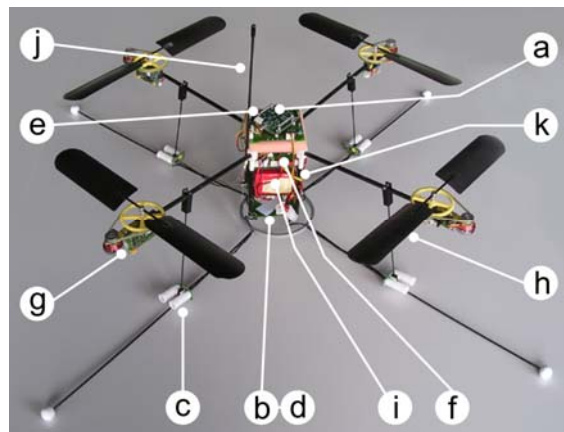
Four ultrasound range finders are mounted on OS4 for obstacle detection, one under each propeller (see Fig. 6.12). Two short plastic tubes are mounted on each sensor in order to reduce the beam width.



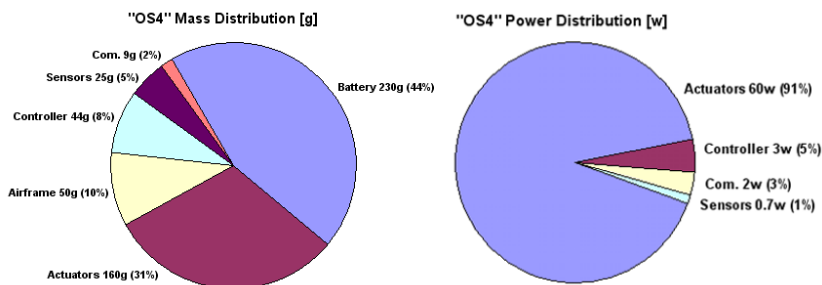
**Fig.6.12.** Possible US sensors arrangement on OS4. Position (3) was adopted after various testing

### Design Results

The robot mass and power distributions are shown in Fig. 6.14. The total mass in the initial design was about 520g, where the battery takes almost one-half and the actuators only one-third thanks to BLDC technology. All the actuators obviously take the lion's part, 60W of 66W average power consumption. However, the latter depends on flight conditions and represents a weighted average between the equilibrium (40W) and the worst possible inclination state (120W) without losing altitude. Fig. 6.13 shows the real robot.



**Fig. 6.13.** Sensors, actuators and electronics of OS4. (a) inertial measurement unit, (b) altitude sensor below the robot, (c) obstacle avoidance sensor with tubes, (d) mini camera below the robot, (e) DSP, (f) mother board, (g) motor module, (h) propeller, (i) battery, (j) RC antenna, (k) wifi dongle



**Fig.6.14.** Mass and power distributions of OS4. The battery mass represents almost one half of the total mass and the actuators sink 90% of the power

## Simulation and Control

The performances expected from the new generation of MFRs can only be achieved through a development of specific control techniques which are likely to deal with the technical limitations especially of sensors and actuators. During the ASL-MFR project we explored several control approaches from theoretical development to final experiments. As a first attempt, we tested on OS4 two linear controllers, a PID and an LQR based on a simplified model. The main result was an autonomous hover flight presented in (Bouabdallah 2004). However, strong disturbances were poorly rejected as in presence of wind. In the second attempt we reinforced the control using backstepping techniques. This time, we were able to elegantly reject strong disturbances but the stabilization in hover flight was delicate (Bouabdallah and Siegwart 2005). Another improvement is now introduced thanks to integral backstepping (Krstić et al. 1995). The idea of using integral action in the backstepping design was first proposed in (Kanellakopoulos et al. 1993) and applied in (Tan et al. 2000) from which this control design was derived. Thanks to this technique, OS4 is able to perform autonomous hovering with altitude control and autonomous take-off and landing.

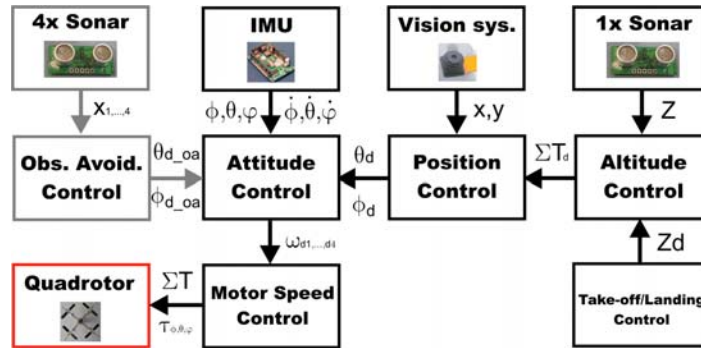


Fig.6.15. The control structure implemented on OS4

The OS4 controller is structured in six different controllers as illustrated in Fig. 6.15. Take-off and landing controller outputs the desired altitude ( $z_d$ ) to altitude controller which outputs the desired overall thrust ( $T_d$ ) based on sonar data. Position controller receives OS4 position ( $x, y$ ) and desired thrust, it outputs desired roll ( $\phi_d$ ) and pitch ( $\theta_d$ ) while desired yaw ( $\psi_d$ ) comes directly from the user. Attitude controller outputs then the desired motor speed to the motor controllers. Integral backstepping

technique is used for attitude, altitude and position control. This permits a powerful and flexible control structure.

### Modelling for Control

The model Eq. (6.8) developed before describes the differential equations of motion of the system. It is advisable for control design to simplify the model in order to comply with the real-time constraints of the embedded control loop. Hence, hub forces and rolling moments are neglected and thrust and drag coefficients are supposed constant. The system can be re-written in state-space form  $\dot{X} = f(X, U)$  with  $U$  inputs vector and  $X$  state vector chosen as follows:

State vector:

$$X = [\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ z \ \dot{z} \ x \ \dot{x} \ y \ \dot{y}]^T \quad (6.10)$$

Inputs vector:

$$U = [U_1 \ U_2 \ U_3 \ U_4]^T \quad (6.11)$$

Where the inputs are mapped by:

$$\begin{cases} U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 = b(-\Omega_2^2 + \Omega_4^2) \\ U_3 = b(\Omega_1^2 - \Omega_3^2) \\ U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{cases} \quad (6.12)$$

The transformation matrix between the rate of change of the orientation angles  $(\dot{\phi} \ \dot{\theta} \ \dot{\psi})$  and the body angular velocities  $(p \ q \ r)$  can be considered as unity matrix if the perturbations from hover are small. Then, one can write  $(\dot{\phi} \ \dot{\theta} \ \dot{\psi}) \approx (p \ q \ r)$ . Simulation tests have shown that this assumption is reasonable. From Eq. (6.8), Eq. (6.10) and Eq. (6.11) we obtain:



$$f(X,U) = \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\psi a_1 + \dot{\theta}a_2\Omega_r + b_1U_2 \\ \dot{\theta} \\ \dot{\phi}\psi a_3 + \dot{\phi}a_4\Omega_r + b_2U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}a_5 + b_3U_4 \\ \dot{z} \\ -g + (\cos\phi\cos\theta)\frac{1}{m}U_1 \\ \dot{x} \\ u_x\frac{1}{m}U_1 \\ \dot{y} \\ u_y\frac{1}{m}U_1 \end{pmatrix} \quad (6.13)$$

$$\begin{pmatrix} a_1 = (I_{yy} - I_{zz})/I_{xx} \\ a_2 = -J_r/I_{xx} \\ a_3 = (I_{zz} - I_{xx})/I_{yy} \\ a_4 = J_r/I_{yy} \\ a_5 = (I_{xx} - I_{yy})/I_{zz} \\ b_1 = l/I_{xx} \\ b_2 = l/I_{yy} \\ b_3 = l/I_{zz} \end{pmatrix} \quad (6.14)$$

$$\begin{cases} u_x = (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ u_y = (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \end{cases} \quad (6.15)$$

It is worthwhile to note in the latter system that the angles and their time derivatives do not depend on translation components. On the other hand, the translations depend on the angles. We can ideally imagine the overall system described by Eq. (6.13) as constituted of two subsystems, the angular rotations and the linear translations. The derivation is similar for attitude, altitude and position controllers. So, only roll angle controller derivation will be presented.

### Attitude Control

Attitude control is the heart of the control system; it keeps the 3D orientation of the helicopter to the desired value. Usually roll and pitch angles are

forced to zero which permits hovering flight. Attitude control loop runs at 76Hz which is the update rate of the IMU<sup>2</sup> (Microstrain 3DM-GX1). The latter provides the rates of turn and orientations around  $(x, y, z)$  axes with an accuracy of  $\pm 2^\circ$  in dynamic. The first step in IB<sup>3</sup> control design is to consider the tracking-error  $e_1 = \phi_d - \phi$  and its dynamics:

$$\frac{de_1}{dt} = \dot{\phi}_d - \omega_x \quad (6.16)$$

The angular speed  $\omega_x$  is not our control input and has its own dynamics. So, we set for it a desired behavior and we consider it as our virtual control:

$$\omega_{xd} = c_1 e_1 + \dot{\phi}_d + \lambda_1 \chi_1 \quad (6.17)$$

$c_1$  and  $\lambda_1$  are positive constants and  $\chi_1 = \int_0^t e_1(\tau) d\tau$  the integral of roll tracking error. So, the integral term is now introduced in Eq. (6.17). Since  $\omega_x$  has its own error  $e_2$ , we compute its dynamics using Eq. (6.17) as follows:

$$\frac{de_2}{dt} = c_1(\dot{\phi}_d - \omega_x) + \ddot{\phi}_d + \lambda_1 e_1 - \ddot{\phi} \quad (6.18)$$

where  $e_2$ , the angular velocity tracking error is defined by:

$$e_2 = \omega_{xd} - \omega_x \quad (6.19)$$

Using Eq. (6.17) and Eq. (6.19) we rewrite roll tracking error dynamics as:

$$\frac{de_1}{dt} = -c_1 e_1 - \lambda_1 \chi_1 + e_2 \quad (6.20)$$

By replacing  $\ddot{\phi}$  in Eq. (6.18) by its corresponding expression from model (6.13), the control input  $U_2$  appears in Eq. (6.21):

$$\frac{de_2}{dt} = c_1(\dot{\phi}_d - \omega_x) + \ddot{\phi}_d + \lambda_1 e_1 - \dot{\theta}\dot{\psi}a_1 - \dot{\theta}a_2\Omega_r - b_1 U_2 \quad (6.21)$$

---

<sup>2</sup> Inertial Measurement Unit

<sup>3</sup> Integral Backstepping

The real control input has appeared in Eq. (6.21). So, using equations Eq. (6.16), Eq. (6.20) and Eq. (6.21) we combine the tracking errors of the position  $e_1$ , of the angular speed  $e_2$  and of the integral position tracking error  $\chi_1$  to obtain:

$$\frac{de_2}{dt} = c_1(-c_1e_1 - \lambda_1\chi_1 + e_2) + \ddot{\phi}_d + \lambda_1e_1 - \tau_x/I_{xx} \quad (6.22)$$

where  $\tau_x$  is the overall rolling torque. The desirable dynamics for the angular speed tracking error is:

$$\frac{de_2}{dt} = -c_2e_2 - e_1 \quad (6.23)$$

This is obtained if one chooses the control input  $U_2$  as:

$$U_2 = \frac{1}{b_1}[(1 - c_1^2 + \lambda_1)e_1 + (c_1 + c_2)e_2 - c_1\lambda_1\chi_1 + \ddot{\phi}_d - \dot{\theta}\dot{\psi}a_1 - \dot{\theta}a_2\Omega_r] \quad (6.24)$$

where  $c_2$  is a positive constant which determines the convergence speed of the angular speed loop. Similarly, pitch and yaw controls are:

$$\begin{cases} U_3 = \frac{1}{b_2}[(1 - c_3^2 + \lambda_2)e_3 + (c_3 + c_4)e_4 - c_3\lambda_2\chi_2 + \ddot{\theta}_d - \dot{\phi}\dot{\psi}a_3 + \dot{\phi}a_4\Omega_r] \\ U_4 = \frac{1}{b_3}[(1 - c_5^2 + \lambda_3)e_5 + (c_5 + c_6)e_6 - c_5\lambda_3\chi_3] \end{cases} \quad (6.25)$$

with  $(c_3, c_4, c_5, c_6, \lambda_2, \lambda_3) > 0$ , and  $(\chi_2, \chi_3)$  the integral position tracking error of pitch and yaw angles respectively.

### **Stability Analysis**

Stability analysis is performed using Lyapunov theory. The following candidate Lyapunov function is chosen:

$$V = \lambda \frac{1}{2} \chi_1^2 + \frac{1}{2} e_1^2 + \frac{1}{2} e_2^2 \quad (6.26)$$

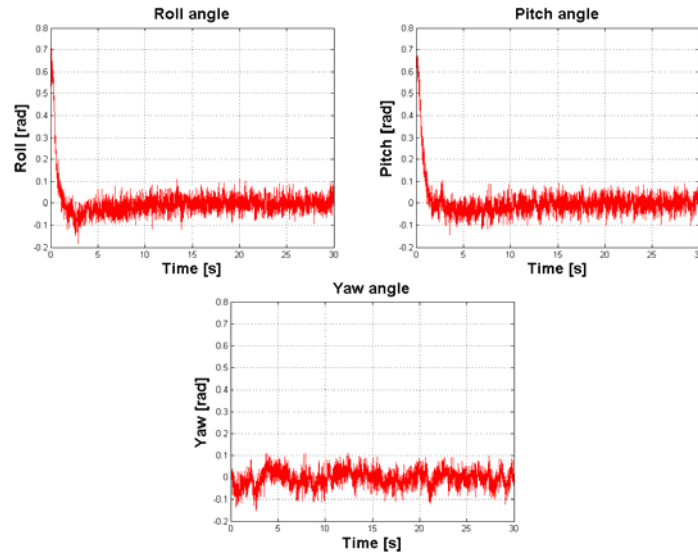
It includes the position tracking error  $e_1$ , its integration  $\chi_1$  and velocity tracking error  $e_2$ . Deriving Eq. (6.26) and using equations Eq. (6.20) and Eq. (6.23) gives:

$$\dot{V} = -c_1 e_1^2 - c_2 e_2^2 \leq 0 \quad (6.27)$$

The definition of Eq. (6.26) and the fact that  $\dot{V} \leq 0, \forall (e_1, e_2)$  guarantees the boundedness of  $e_1$ ,  $\chi_1$  and  $e_2$ . The desired position reference  $\phi_d$  is bounded by assumption and  $e_1 = \phi_d - \phi$  is also bounded, so, position  $\phi$  is also bounded. This implies the boundedness of the virtual control  $\omega_x$ . Finally, the boundedness of the overall control torque is due to our choice of the control law in Eq. (6.24). The system is also globally asymptotically stable from the positive definition of  $V$  and the fact that  $\dot{V}(e_1, e_2) < 0, \forall (e_1, e_2) \neq 0$  and  $\dot{V}(0) = 0$ .

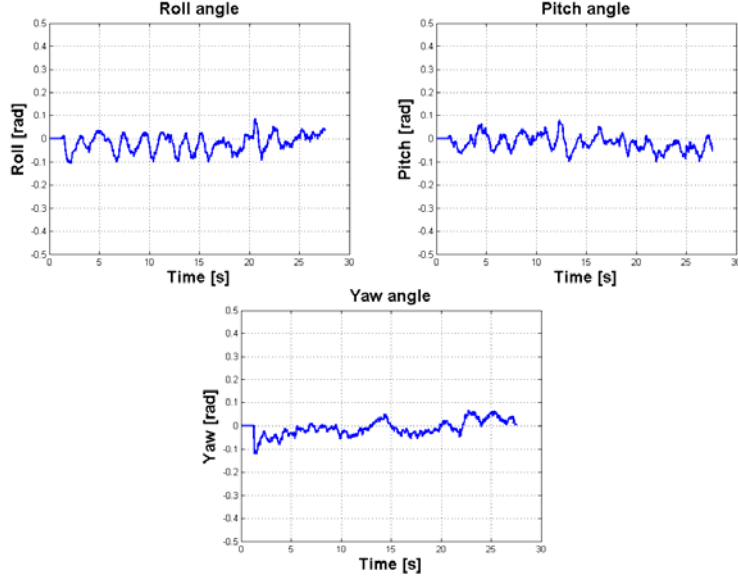
### **Results**

Attitude control performance is of crucial importance, it is directly linked to the performance of the actuators. OS4 is equipped with brushless sensorless motors powerful enough to avoid amplitude saturation. However, they suffer from low dynamics and thus from bandwidth saturation. This was taken into account in control design. Simulation results shown in Fig. 6.16 are performed with a model which includes actuators' dynamics and amplitude saturation. The simulation takes into account the delay and the noise inherent to sensors. The task was to stabilize roll, pitch and yaw angles and maintain them at zero. Control parameters were in the simulation  $C_1 = 10, C_2 = 2, C_3 = 10, C_4 = 2, C_5 = 2, C_6 = 2$ .



**Fig.6.16.** Simulation: Integral backstepping attitude controller has to maintain roll, pitch and yaw angles to zero. Despite of the hard initial conditions and the white noise, the helicopter is quickly brought back to equilibrium

The experiment shown in Fig. 6.17. is a free flight where attitude references are zero. One can see in roll and pitch plots a bounded oscillation of 0.1rad in amplitude. This oscillation is not perceptible in flight; nevertheless it is due to the slow dynamics of OS4's actuators coupled with the differences between the four propulsion groups. Control parameters were in this experiment  $C_1 = 10.5$ ,  $C_2 = 2$ ,  $C_3 = 10$ ,  $C_4 = 2$ ,  $C_5 = 2$ ,  $C_6 = 2$ . These are really close to the parameters used in simulation which highlights the quality of the model.



**Fig.6.17.** Experiment: Integral backstepping attitude controller has to maintain attitude angles to zero in flight. The helicopter is stabilized despite the numerous disturbances due to yaw drift, sensors noise and unmodeled effects

### Altitude Control

The altitude controller keeps the distance of the helicopter to the ground at a desired value. It is based on a sonar (Devantech SRF10) which gives the range to the closest obstacle at 15 Hz. The accuracy depends on the distance, it is about 1 to 2 cm at 1 m. The necessary altitude rate of change is estimated based on the range. On the control law side, altitude tracking error and the speed tracking error are defined as:

$$e_7 = z_d - z \quad (6.28)$$

$$e_8 = c_7 e_7 + \dot{z}_d + \lambda_4 \chi_4 - \dot{z} \quad (6.29)$$

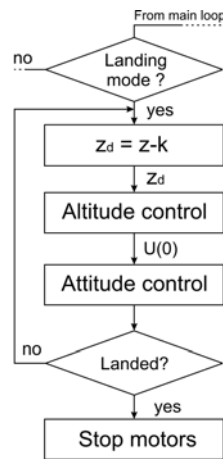
The control law is then:

$$U_1 = \frac{m}{\cos\phi\cos\theta} [g + (1 - c_7^2 + \lambda_4) e_7 + (c_7 + c_8) e_8 - c_7 \lambda_4 \chi_4] \quad (6.30)$$

where  $(c_7, c_8, \lambda_4)$  are positive constants.

### Take-off and Landing

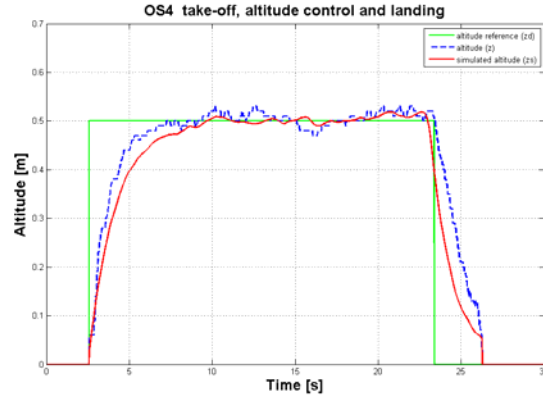
The autonomous take-off and landing algorithm adapts the altitude reference  $z_d$  to follow the dynamics of the quadrotor for taking-off or landing. One can see in Fig. 6.18. that the desired altitude reference is gradually reduced by a fixed step  $k$  ( $k > 0$ ) which depends on the vehicle dynamics and the desired landing speed. Moreover, the fact that the control loop is much faster than the vehicle dynamics makes the landing very smooth. Ground effect was not implemented because the landing skids are long enough to keep the propellers out of ground effect even after touch-down.



**Fig.6.18.** Autonomous landing flowchart. Altitude reference is gradually reduced taking into account the dynamics of the robot

### Results

Altitude control works surprisingly well despite all the limitations of the sonar. Fig. 6.19. shows an altitude reference profile (green) followed by the simulated controller (red) and the real controller (blue). The task was to climb to 0.5 m, hover and then land. Control parameters were  $C_7 = 3.5$ ,  $C_8 = 1.5$  in simulation and  $C_7 = 4$ ,  $C_8 = 2$  in experiment. The slight deviation between simulation and reality in take-off and landing phases is inherited from actuators' dynamics where the model was slightly slower, in the raising edge, and slightly faster in the falling one. Take-off is performed in 2 s (0-0.5 m) and landing in 2.8 s (0.5-0 m). Altitude control has a maximum of 3 cm deviation from the reference.



**Fig.6.19.** Autonomous take-off, altitude control and landing in simulation and in real flight

### Position Control

Position control keeps the helicopter over the desired point. It is meant here the  $(x, y)$  horizontal position with regard to a starting point. Horizontal motion is achieved by orienting the thrust vector towards the desired direction of motion. This is done by rotating the vehicle itself in the case of a quadrotor. In practice, one performs position control by rolling or pitching the helicopter in response to a deviation from the  $y_d$  or  $x_d$  references respectively. Thus, the position controller outputs the attitude references  $\phi_d$  and  $\theta_d$ , which are tracked by the attitude controller (see Fig. 6.15). The thrust vector orientation in the earth fixed frame is given by  $R$ , the rotation matrix. Applying small angle approximation to  $R$  gives:

$$R = \begin{bmatrix} 1 & \psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{bmatrix} \quad (6.31)$$

From Eq. (6.13) and using Eq. (6.31) one can simplify horizontal motion dynamics to:



$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \end{bmatrix} = \begin{bmatrix} -\theta U_1 \\ \phi U_1 \end{bmatrix} \quad (6.32)$$

The control law is then derived using IB technique. Position tracking errors for  $x$  and  $y$  are defined as:

$$\begin{cases} e_9 = x_d - x \\ e_{11} = y_d - y \end{cases} \quad (6.33)$$

Accordingly speed tracking errors are:

$$\begin{cases} e_{10} = c_9 e_9 + \dot{x}_d + \lambda_5 \chi_5 - \dot{x} \\ e_{12} = c_{11} e_{11} + \dot{y}_d + \lambda_6 \chi_6 - \dot{y} \end{cases} \quad (6.34)$$

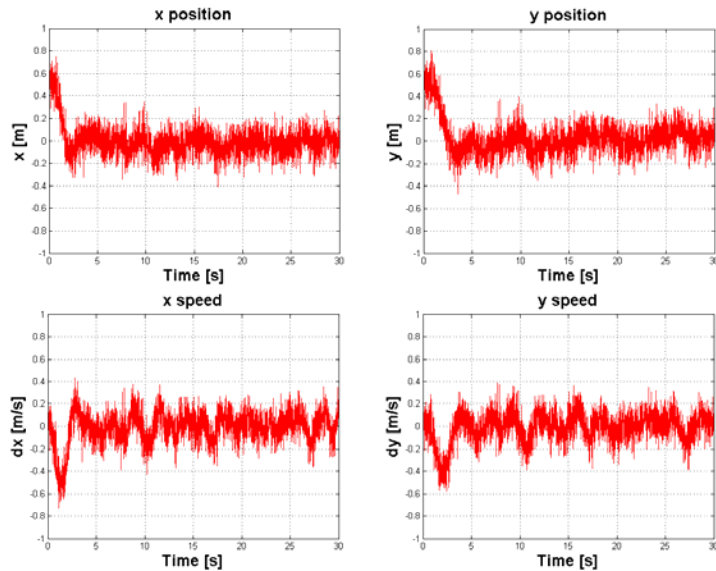
The control laws are then:

$$\begin{cases} U_x = \frac{m}{U_1} [(1 - c_9^2 + \lambda_5) e_9 + (c_9 + c_{10}) e_{10} - c_9 \lambda_5 \chi_5] \\ U_y = -\frac{m}{U_1} [(1 - c_{11}^2 + \lambda_6) e_{11} + (c_{11} + c_{12}) e_{12} - c_{11} \lambda_6 \chi_6] \end{cases} \quad (6.35)$$

where  $(c_9, c_{10}, c_{11}, c_{12}, \lambda_5, \lambda_6)$  are positive constants.

### **Results**

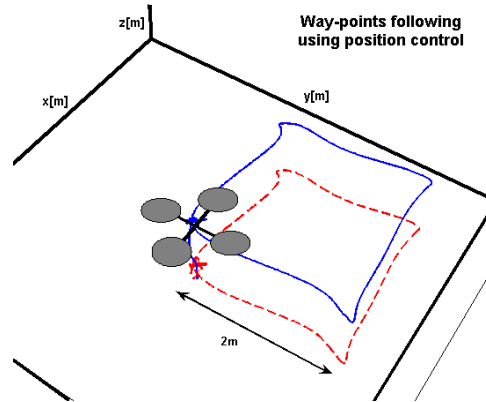
The main result in position control was obtained in simulation. Fig. 6.15 shows how the different controllers are cascaded. In fact, only the attitude is driven by the position, altitude controller is simply feeding them with  $U_1$ . Attitude and position loops run at 76 Hz and 25 Hz respectively. This spectral separation is necessary to avoid a conflict between the two loops; it is often accompanied with gain reductions in the driving loop. Control parameters were  $C_9 = 2, C_{10} = 0.5, C_{11} = 2, C_{12} = 0.5$  in the simulation of Fig. 6.20



**Fig.6.20.** Simulation: Integral backstepping position controller drives attitude controller in order to maintain the helicopter over a given point

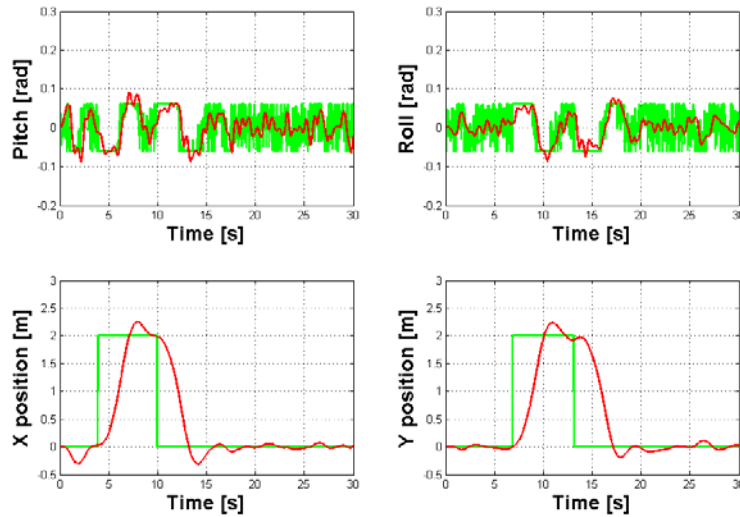
### ***Way-points following***

The planner block in Fig. 6.4 defines the way-points and hence the trajectories OS4 has to follow. The position of the next way-point is sent to position controller which directs the vehicle towards the goal. A way-point is declared reached when the helicopter enters a sphere around this point. The radius of this sphere (0.1 m) is the maximum admitted error. Fig. 6.21 shows a square trajectory defined by four way-points. The task was to climb to 1 m from the ground and then follow the four way-points of a square of 2 m side.



**Fig.6.21.** Four way-points for a square trajectory tracked by OS4

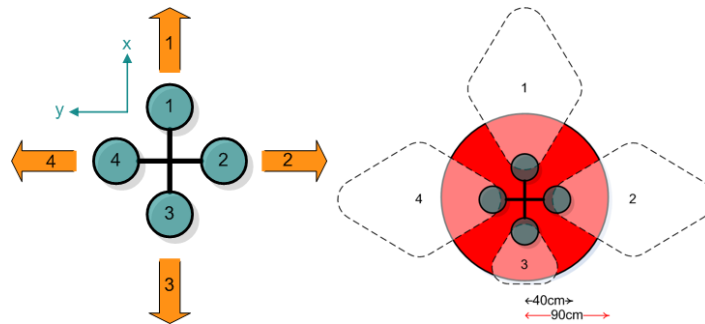
In order to track the square trajectory, the planner generates the  $(x_d, y_d)$  position references, and consequently the position controller generates the  $(\phi_d, \theta_d)$  attitude references. Fig. 6.22 depicts these signals and shows that the 2m side square is tracked with about 10% overshoot (20cm), while the trajectory is completed in 20s.



**Fig.6.22.** Simulation: The position and attitude signals generated to track the square trajectory

## Obstacle avoidance

OS4 is equipped with a sonar-based obstacle avoidance system composed of four miniature ultrasound range finders in cross configuration. First of all, we introduced the obstacle avoidance controller into the Simulink model and inserted the environment and sensor libraries. Aiming at simplify the procedure; we decided to keep the altitude constant during evasive maneuvers. This would reduce the path planning complexity to a 2D problem. We also restricted its direction of flight: OS4 can move only on the four directions where the US sensors were placed. To increase the flight safety, a 90 cm radius security zone is constantly maintained between the helicopter and the environment (see Fig. 6.23). This zone assures a 50 cm distance between the helicopter rotors and any obstacle. If an obstacle is detected inside the security zone, a safety loop (that runs in parallel to the OAC<sup>4</sup>) interferes in the helicopter flight control and generates an evasive maneuver. This maneuver is obtained by selecting a predefined pitch and/or roll angles that would avoid a collision between the helicopter and the obstacles.



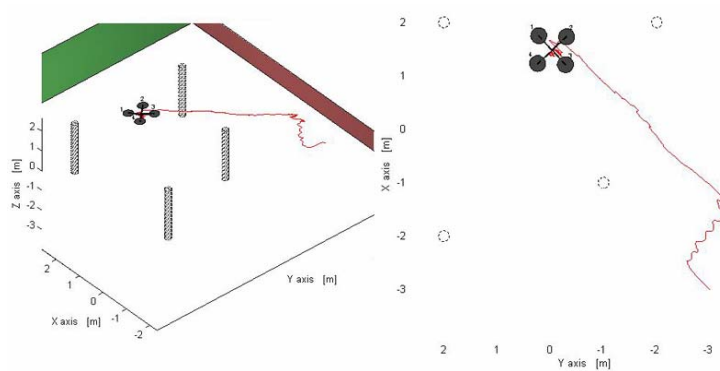
**Fig.6.23.** The 4 flight directions (left) and the security zone (right)

Several algorithms were simulated with various results in a  $100 \text{ m}^2$  environment with obstacles represented as columns of 20 cm in diameter and 3 m in height.

The developed approaches can be divided into two categories: Relative position and speed-based approaches. The first OAC uses position controller to act on the relative position of the helicopter with regard to the closest obstacle  $(X_{oa}, Y_{oa})$ . The second one uses speed controller to act on the speed of the vehicle  $(\dot{X}, \dot{Y})$  if an obstacle is detected. The latter approach

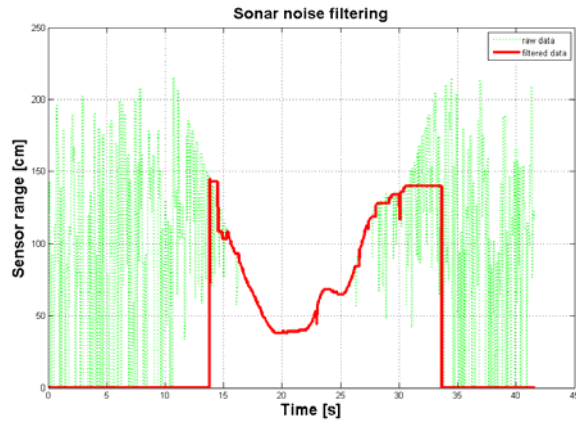
<sup>4</sup> Obstacle Avoidance Controller

was used to develop our main OAC algorithm. The idea was to act on  $\dot{X}$  and  $\dot{Y}$  while keeping the heading and altitude constants. When an obstacle is detected its distance to the helicopter is classified based on a given threshold as "far", "close" or "too close". If the obstacle is "far", no avoidance action is needed and the OAC does not interfere with the helicopter normal flight. On the other hand, if the obstacle distance is "close" or "too close" the OAC informs the helicopter flight control, reduces its speed, and generates evasive maneuvers using predefined flight directions, this is shown in Fig. 6.24. The selection of the direction of the evasive flight depends on the stimulated sensor and the desired flight direction previously selected by the user. However, if the quadrotor is surrounded by obstacles that are "too close", it reduces the speed and keeps a hovering behavior.



**Fig.6.24.** Simulation: OS4 avoiding static obstacles

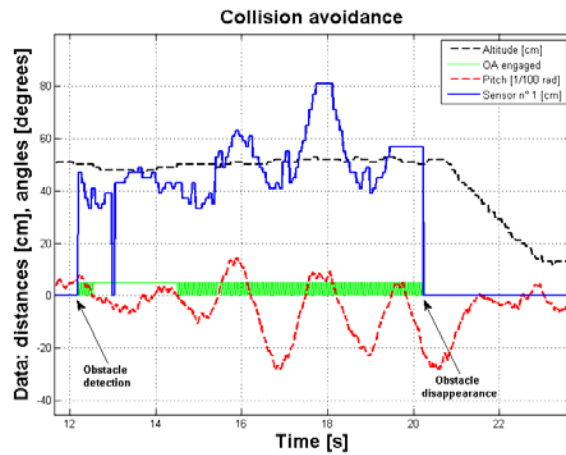
The lack of precise sensors for linear speed made the implementation of this approach difficult. A simple collision avoidance algorithm was then developed. The idea is to avoid collision with walls or persons present in the flight area. The inherent noise of the sonar especially in absence of obstacles was threatening OS4 stability. This is mainly due to the interferences between the five sonars and the effect of the propellers on the ultrasound waves. Fig. 6.25 shows a detection of an obstacle with and without the filter. The latter is based on the variation of successive samples and gives a reliable detection signal usable in flight.



**Fig.6.25.** Obstacle detection with and without the filter

### Results

A collision avoidance behavior was practically obtained after numerous tests and tuning. Once the obstacle is detected, a pitch reference is given to fly away the helicopter from the obstacle. Fig. 6.26 shows the reaction of OS4 to an obstacle at 40 cm, one can see the distance to the obstacle increasing until the latter disappears, then OS4 recovers a normal flight.



**Fig.6.26.** Experiment: Collision avoidance with OS4. The helicopter flies back until the obstacle disappears

## Conclusion and Future Work

This chapter presented the latest developments in ASL-MFR project. A quadrotor simulation model was introduced. It includes major aerodynamical effects modelled with blade element and momentum theory. In addition, the actuator's model was identified and all sensor delays and noises were taken into account. Moreover, a Matlab/Simulink based simulator was developed for testing the controllers. Real experiments were conducted with the same control parameters tuned in simulation. In the second part of this chapter we introduced a practical methodology for miniature rotorcraft design. It was the only tool used to design the helicopter "OS4" achieving 100% thrust margin for 30 min autonomy (in the initial design). The last version of OS4 embeds all the necessary avionics and energy devices for a fully autonomous flight. This comprises a low cost IMU, a vision based position sensor specifically developed for this project and an obstacle detection setup. The third part introduced the control approach proposed which permitted the design of the main controllers: Attitude, altitude, position and the auxiliary ones: Take-off and landing, obstacle avoidance and way-point following. The latter was demonstrated in simulation. The experiment has shown that OS4 is currently able to take-off, hover, land and avoid collisions automatically thanks to model-based control. The future work is to firstly enhance the propulsion group towards more reliability and high bandwidth. Secondly, it is necessary to improve the vision sensor in order to get rid of the external pattern and to provide stable vision-based yaw information. It is also possible to reduce the mass of the helicopter by using the tiny single board computers available now. Thirdly, it would be very interesting to practically test a way-points following maneuver with obstacle avoidance capability. OS4 is undoubtedly one of the most integrated quadrotor ever designed. As far as we know, it is the first one practically capable of a collision avoidance maneuver.

## References

AERO-EPFL, <http://aero.epfl.ch/>.

Bouabdallah S *et al.* (2004) PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor. In Proc. (IEEE) International Conference on Intelligent Robots and Systems (IROS'04), Sendai, Japan

Bouabdallah S and Siegwart R (2005) Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In Proc. (IEEE) International Conference on Robotics and Automation (ICRA'05), Barcelona, Spain

Bouabdallah S *et al.* (2007) Autonomous miniature flying robots: Coming soon!," *Robotics and Automation Magazine*

Cheeseman I and Bennett W (1957) The effect of the ground on a helicopter rotor in forward flight. *Aeronautical Research Council*, no. 3021

DeMenthon D and Davis L (2005) Model-based object pose in 25 lines of code *International Journal of Computer Vision*, vol. 15

Deng X, et al. (2003) Attitude control for a micromechanical flying insect including thorax and sensor models. In Proc. (IEEE) International Conference on Robotics and Automation (ICRA'03), Taipei, Taiwan

Fay G, (2001) Derivation of the aerodynamic forces for the mesicopter simulation (Derivation of the Aerodynamic Forces for the Mesicopter Simulation, Stanford University)

Griffiths D and Leishman J (2002) A study of dual-rotor interference and ground effect using a free-vortex wake model. In *Proc. of the 58th Annual Forum of the American Helicopter Society*, Montréal, Canada

Guenard N *et al.*, (2006) Control laws for the tele operation of an unmanned aerial vehicle known as an x4-flyer. In *Proc. (IEEE) International Conference on Intelligent Robots (IROS'06)*, Beijing, China

Hoffmann G, et al. (2004) The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In 23rd Digital Avionics Systems Conference (DASC'04), Salt Lake City, USA.



- Kanellakopoulos I and Krein P, (1993) Integral-action nonlinear control of induction motors,” in *Proc. of the 12th IFAC World Congress*, Sydney, Australia
- Krstić M *et al.* (1995) *Nonlinear and Adaptive Control Design*. New York, USA: Wiley Interscience
- Kroo I, *et al.* (2000) The mesicopter: A miniature rotorcraft concept phase 2 interim report.
- Leishman J G *Principles of Helicopter Aerodynamics*. Cambridge University Press, Cambridge
- MARV, <http://www.enaе.umd.edu/AGRC/Design00/MARV.html>.
- Murray R, *et al.* (1994) *A Mathematical Introduction to Robotic Manipulation*. CRC, Boca Raton
- Wang W *et al.* (2006) Autonomous control for micro-flying robot and small wireless helicopter x.r.b. In *Proc. (IEEE) International Conference on Intelligent Robots (IROS'06)*, Beijing, China
- Nicoud J D and Zufferey J C (2002) Toward indoor flying robots. In *Proc. (IEEE) International Conference on Intelligent Robots (IROS'02)*, Lausanne, Switzerland
- OpenCV, <http://www.intel.com/research/mrl/research/opencv/>
- Tan Y *et al.* (2000) Advanced nonlinear control strategy for motion control systems,” in *Proc. of (IEEE) Power Electronics and Motion Control Conference, (PIEMC'00)*, Beijing, China