Control of the Mascarillon robot: a 4-DOF, cube-shaped flying robot

June 2007

Stefan Bracher

Abstract:

Three different types of controllers, fuzzy control, sliding mode control and pre-calculated couple control, have been developed and implemented on a 4 degree of freedom flying robot called “Mascarillon”. Two of the three modes are designed to be adaptive, in order to account for the non-linear and changing dynamics of the robot. This paper presents the different controllers and shows a performance comparison, based on experimental results.

Keywords: Non-linear Dynamics, fuzzy control, sliding mode control, pre-calculated couple, adaptive control

A. Introduction

1. The robot

The Mascarillon robot is a cube shaped helium blimp robot. Equipped with fans in 8 of the 12 cube corners, it can move in all directions and rotate around the vertical axis. Sonar sensors and an electronic compass allow to compute the position and orientation relative to the environment. The core of the robot is a “Korebot” [1] Linux system, on which the positioning and behavioral control algorithms are programmed.

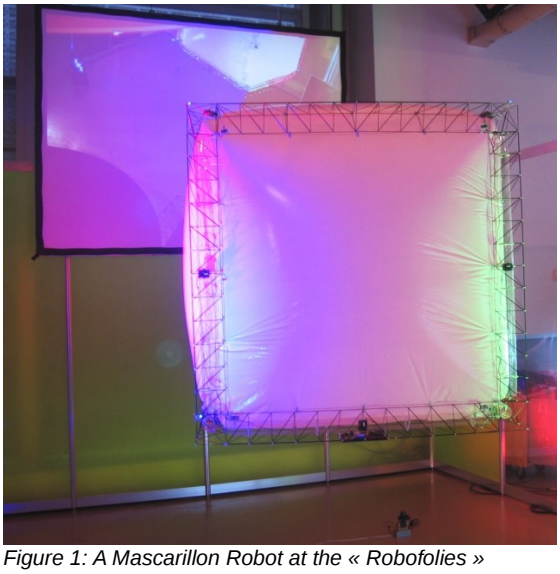

Figure 1: A Mascarillon Robot at the « Robofolies » exhibition, Centre de Science de Montreal, March 2008

2. Purpose of the robot

The Mascarillons have been produced for the SAILS [2] project, in order to produce self assembling lighter than air structures for architectural research. In simulation, the necessary swarm-intelligence protocols are already tested, but they are not applied yet to the real cubes.
Currently the cubes are used for public performances, where they interact with an actor or the audience. Due to the cube shape, the cubes can also be used as a flying projection surface.

3. Main issues

The most complex task for the robot is just to keep its position. Because of his low mass and cubic shape, it is moved by the slightest air currents. Sun radiation easily heats up the helium such as the blimp looses its equilibrium. Further on, different helium contents and battery power change its dynamics constantly.

A first attempt with simple PID control was able to stabilize the system, but proved not to be practical as the controller gains had to be experimentally fine-tuned for each presentation setup. As no adequate model, able to accurately reproduce the real robots behavior could be found, it was decided to implement and test different controller types directly on the real robot itself.

B. Control

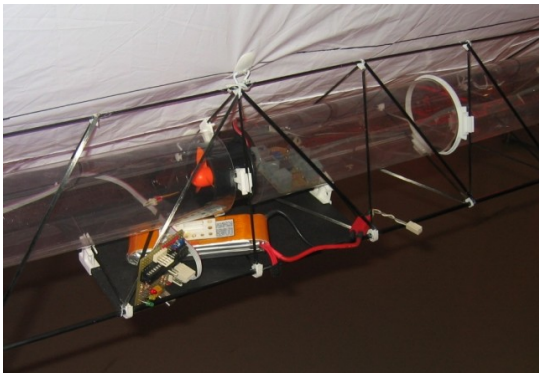

Figure 2: Fan used for propulsion in one of the corners

1. The selected controller types

The pre-calculated couple, the adaptive sliding mode control and the fuzzy control have been selected as they all offer a solution to the non-exact model problem.
While fuzzy control is based on control laws that are similar to the ones of a human operator and do not require a model at all, pre-calculated couple and sliding-mode control separate the actual control law from the model. The model is then optimized on-line with appropriate adaptation algorithms. Details will follow in section three.

2. Coordinate system and approximative dynamic model

The coordinate system is placed in the center of mass of the robot. The coordinates represent the distance to the walls surrounding the robot. For each axis, it is decided in the control program, which side is taken into consideration. Distance measurements taken on the back, left and down side of the robot are regarded as positive coordinates, on the front, right and up side as negative coordinates.
The rotational angle around the z axis is measured with a compass, relative to the initial robot orientation. It is positive, if the robot has been turned clockwise negative otherwise.

The four degrees of freedom of the robot are the lateral movements (displacement x, y, z) and the rotation around the z-axis (Angle Θ_z, just referred as Θ). The robot can theoretically also turn around the other two axes (Angles Θ_x, Θ_y). The available motor power however is unable to turn the “center of mass”, which is situated below the geometrical cube center, around these axes. Thus this movement is not actively controlled and the angles Θ_x, Θ_y are neglected.

The approximative dynamic model of the robot is:

$$\mathbf{F} = \begin{bmatrix} F_x \\ F_y \\ F_z \\ \tau_z \end{bmatrix} = \begin{bmatrix} m \cdot \ddot{x} + k_{lat} \cdot \dot{x}^2 \cdot sign(\dot{x}) \\ m \cdot \ddot{y} + k_{lat} \cdot \dot{y}^2 \cdot sign(\dot{y}) \\ m \cdot \ddot{z} + k_{lat} \cdot \dot{z}^2 \cdot sign(\dot{z}) \\ I_z \cdot \ddot{\Theta} + k_{rot} \cdot \dot{\Theta}^2 \cdot sign(\dot{\theta}) \end{bmatrix} \qquad \textit{Eq. B.2.1}$$

with

F_x, F_y, F_z : Forces in x, y, and z direction applied by the fans

τ_z : Torque around z axis applied by the fans

k_{lat}, k_{rot} : Aerodynamic resistance to lateral movement and rotation

I_z : Cube inertia around z axis

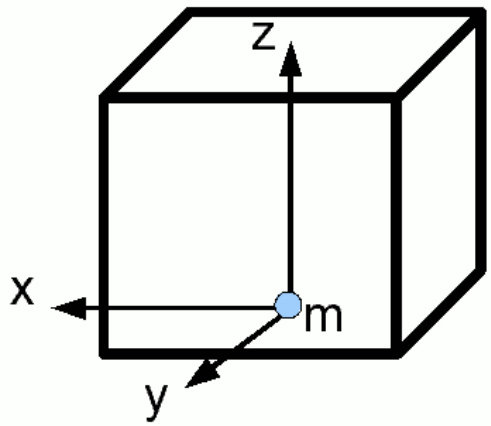

Figure 3: The axes of the coordinate system intersect in the center of mass of the Mascarillon. Note that the center of mass is situated below the geometrical center.

Equation B.2.1 can be written in short form as

$$\mathbf{F} = \mathbf{Y} \cdot \mathbf{a} \qquad \textit{Eq. B.2.2}$$

with $\mathbf{Y} = \begin{bmatrix} \ddot{x} & \dot{x}^2 \cdot sign(\dot{x}) & 0 & 0 \\ \ddot{y} & \dot{y}^2 \cdot sign(\dot{y}) & 0 & 0 \\ \ddot{z} & \dot{z}^2 \cdot sign(\dot{z}) & 0 & 0 \\ 0 & 0 & \ddot{\Theta} & \dot{\Theta}^2 \cdot sign(\dot{\Theta}) \end{bmatrix}$ and $\mathbf{a} = \begin{bmatrix} m \\ k_{lat} \\ I_z \\ k_{rot} \end{bmatrix}$

3. The controllers in detail

3.1. Fuzzy Control

Fuzzy Control follows a set of rules instead of mathematical functions. The advantage is that no model is needed and that an intuitive control scheme, based on human experience, is resulting. The disadvantage of this approach is the need for experimental fine tuning of the controller parameters.

In order that these these "fuzzy" rules can be executed by a controller, they have to be quantized. This is done by selecting specific system states and connecting those states to a certain action.

For each degree of freedom $\mathbf{x}$, the following states, based on the error (difference between desired and actual position), are selected:

- Positive error : Z_{ep}
- Negative error : Z_{en}
- Integral of error positive : Z_{intep}
- Integral of error negative : Z_{inten}
- Absolute error increasing : Z_{ince}
- Absolute error decreasing: Z_{dece}
- Error slightly positive : Z_{esp}
- Error slightly negative : Z_{esn}

The system can belong to each of this states Z_i with a membership σ_i between 0 and 1. The membership is calculated with the membership functions on the right.

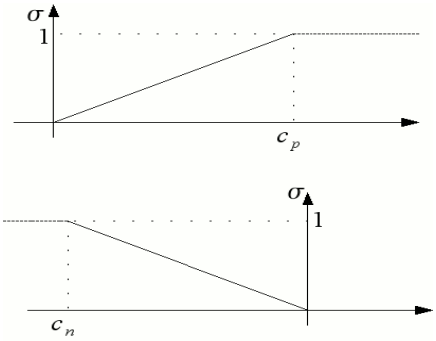

Figure 4: Top: Membership function used for states: "Positive error", "Integral of error positive" and "absolute error increasing"". Bottom: Membership function used for states: "negative error", "Integral of error negative" and "absolute error decreasing". With the error, the integral of the error and the absolute of the first derivative of the error as x-axis and experimental parameters c_p, c_n

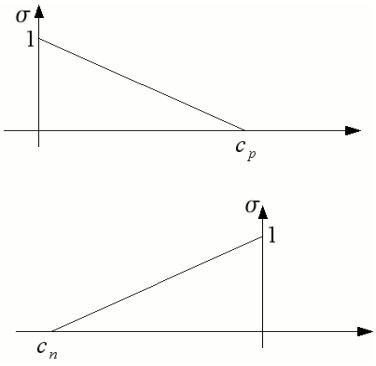

Figure 5: Top: Membership function for state "Error slightly positive". Bottom: Membership for state "Error. With the error as x-axis and experimental parameters c_p, c_n

Set of rules for each degree of freedom $\mathbf{x}$ *:*

State	Action:	Explanation
Z_{ep}	Apply medium positive force Γ_m	The actual position coordinate is smaller than the desired one.
Z_{en}	Apply medium negative force $-\Gamma_m$	The actual position coordinate is bigger than the desired one.
Z_{ep} and Z_{intep}	Apply big positive force Γ_p	For a while, the actual position coordinate is smaller than the desired one.
Z_{en} and Z_{inten}	Apply big negative force $-\Gamma_p$	For a while, the actual position coordinate is bigger than the desired one.
Z_{ep} and Z_{ince}	Apply big positive force Γ_p	The robot goes away from the desired position in negative direction.
Z_{en} and Z_{ince}	Apply big negative force $-\Gamma_p$	The robot goes away from the desired position in positive direction.
Z_{esp} and Z_{dece}	Apply small negative force $-\Gamma_s$ to prevent overshoot	The robot is approaching the desired location from the positive side
Z_{esn} and Z_{dece}	Apply small positive force Γ_s to prevent overshoot	The robot is approaching the desired location from the negative side

The force to be applied for each degree of freedom $\mathbf{x}_i$, according to the above set of rules, is then calculated with the following defuzzification law:

$$F_i = \Gamma_{i,m} \cdot (\sigma_{i,ep} - \sigma_{i,en}) + \Gamma_{i,p} \cdot (\sigma_{i,ep} \cdot (\sigma_{i,intep} + \sigma_{i,ince}) - \sigma_{i,en} \cdot (\sigma_{i,inten} + \sigma_{i,ince})) + \Gamma_{i,s} \cdot (\sigma_{i,esn} \cdot \sigma_{i,dece} - \sigma_{i,esp} \cdot \sigma_{i,dece})$$

Eq. B.3.1

3.2. Sliding-Mode Control

The sliding Sliding mode controller is divided in a kinetic controller and a dynamic conversion part. This makes possible that the actual control law (the kinetic controller) is decoupled from the dynamic model. Its implementation is based on [3].

Kinetic controller

The kinetic controller calculates the velocity $\dot{\mathbf{x}}_{\mathbf{kin}}$, needed to reduce the error between desired and actual position:

$$\dot{\mathbf{x}}_{\mathbf{kin}} = \boldsymbol{\Delta} \cdot (\mathbf{x}_{\mathbf{des}} - \mathbf{x})$$

with the matrix of controller gains $\boldsymbol{\Delta}$

Dynamic conversion

The dynamic conversion of $\dot{\mathbf{x}}_{\mathbf{kin}}$ into the forces required to attain these velocities is done using the dynamic model of the robot (Equation B.2.2):

$$\mathbf{F} = \mathbf{Y} \cdot \mathbf{a}$$

with $\mathbf{Y} = \begin{bmatrix} \ddot{x}_{kin} & \dot{x}_{kin}^2 \cdot sign(\dot{x}_{kin}) & 0 & 0 \\ \ddot{y}_{kin} & \dot{y}_{kin}^2 \cdot sign(\dot{y}_{kin}) & 0 & 0 \\ \ddot{z}_{kin} & \dot{z}_{kin}^2 \cdot sign(\dot{z}_{kin}) & 0 & 0 \\ 0 & 0 & \ddot{\Theta}_{kin} & \dot{\Theta}_{kin}^2 \cdot sign(\dot{\Theta}_{kin}) \end{bmatrix}$ and $\mathbf{a} = \begin{bmatrix} m \\ k_{lat} \\ I_z \\ k_{rot} \end{bmatrix}$

Parameter estimator

A parameter estimator is introduced to adapt the constants used in the model. The adaptation is done with:

$$\mathbf{a} = \int \boldsymbol{\lambda} \cdot \mathbf{Y}^{\mathbf{T}} \cdot (\dot{\mathbf{x}}_{\mathbf{kin}} - \dot{\mathbf{x}}) \, dt$$

with the weighting matrix $\boldsymbol{\lambda} = \begin{bmatrix} \lambda_m & 0 & 0 & 0 \\ 0 & \lambda_{klat} & 0 & 0 \\ 0 & 0 & \lambda_{Iz} & 0 \\ 0 & 0 & 0 & \lambda_{krot} \end{bmatrix}$

This makes it possible, within certain limits, to adjust for modeling errors and changing environmental conditions. The resulting parameters however do not necessarily correspond to the real parameters. If the model contains errors, they will be a best fit solution.

The main problem of the parameter estimator is that jumps in the commanded position may lead to a wrong parameter estimation and that the behavior of the robot changes over time. To reduce the impact of this, the parameters have be born to certain values.

3.3 Pre-calculated couple

The pre-calculeted couple controller is, like the sliding-mode controller, divided in two parts in order to decouple the control law from the dynamic model. The difference is, that not the velocities $\dot{\mathbf{x}}_{\mathbf{kin}}$, but the accelerations $\ddot{\mathbf{x}}_{\mathbf{kin}}$, necessary to reduce the error are computed in the kinetic part. Alike the sliding mode controller, the implementation of the pre-calculated couple controller is based on [3].

Kinetic controller
The kinetic controller is implemented with a traditional PID approach of the form:

$$\ddot{\mathbf{x}}_{\mathbf{kin}} = \mathbf{K_p}(\mathbf{x}_{\mathbf{des}} - \mathbf{x}) + \mathbf{K}_{\mathbf{d}}(\dot{\mathbf{x}}_{\mathbf{des}} - \dot{\mathbf{x}}) + \mathbf{K_i}\int \mathbf{x}_{\mathbf{des}} - \mathbf{x}\, dt$$

with controller gain matrices $\mathbf{K_p} = \begin{bmatrix} Kp_x & 0 & 0 & 0 \\ 0 & Kp_y & 0 & 0 \\ 0 & 0 & Kp_z & 0 \\ 0 & 0 & 0 & Kp_\tau \end{bmatrix}$, $\mathbf{K_d} = \begin{bmatrix} Kd_x & 0 & 0 & 0 \\ 0 & Kd_y & 0 & 0 \\ 0 & 0 & Kd_z & 0 \\ 0 & 0 & 0 & Kd_\tau \end{bmatrix}$ and

$\mathbf{K_i} = \begin{bmatrix} Ki_x & 0 & 0 & 0 \\ 0 & Ki_y & 0 & 0 \\ 0 & 0 & Ki_z & 0 \\ 0 & 0 & 0 & Ki_\tau \end{bmatrix}$

Dynamic conversion
To compute the dynamic conversion, a modified version of Eq. B.2.2 is used, taking in account only the acceleration:

$$\mathbf{F} = \mathbf{Y} \cdot \mathbf{a}$$

with $\mathbf{Y} = \begin{bmatrix} \ddot{x}_{kin} & 0 \\ \ddot{y}_{kin} & 0 \\ \ddot{z}_{kin} & 0 \\ 0 & \ddot{\Theta}_{kin} \end{bmatrix}$ and $\mathbf{a} = \begin{bmatrix} m \\ I_z \end{bmatrix}$

Parameter estimation
The parameter estimator used to adapt the constants of the dynamic conversion is this time of the form:

$$\mathbf{a} = \int \boldsymbol{\lambda} \cdot \mathbf{Y}^{\mathbf{T}} \cdot (\mathbf{x}_{\mathbf{des}} - \mathbf{x})\, dt$$

with the weighting matrix $\boldsymbol{\lambda} = \begin{bmatrix} \lambda_m & 0 \\ 0 & \lambda_I \end{bmatrix}$

Again the values of **a** are born to minimal and maximal settings.

C Experiments

1. Test setup

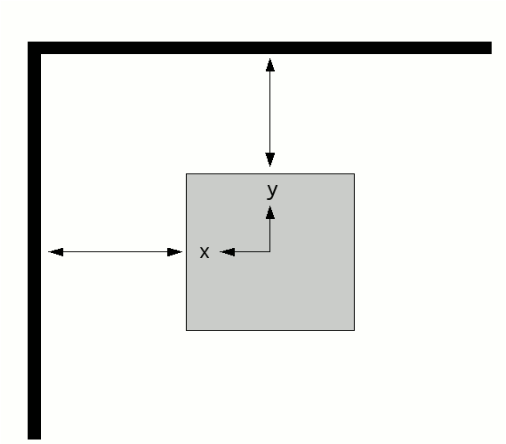

Figure 6: Robot in a room corner: Top view

To test and compare the different controllers, the cube is placed in a corner of a room, with x and y axes parallel to the wall. (See figure 6). As distances are measured on the front, left and down side, in this setup, the x- and y-coordinates are negative while the z-coordinate is positive. (Refer to section B, 2.)
The rotation angle Θ around the z-axis is set to zero with the robot parallel to the walls as in figure 6.

A test sequence is then performed, which consists of the robot:

1. staying at a distance of 1m from each wall, at an altitude of 0.85m and no rotation.
2. increasing the distance to the wall on the x-side to 1.57m
3. increasing the distance to the wall on the y-side to 1.57m
4. increasing the altitude to 1.42m
5. decreasing the distance to the wall on the x-side to 1m
6. decreasing the distance to the wall on the y-side to 1m
7. decreasing the altitude to 0.85m
8. rotating -0.380 rad
9. rotating back to initial orientation

This sequence is repeated several times for all three controllers.

2. Results Fuzzy Logic

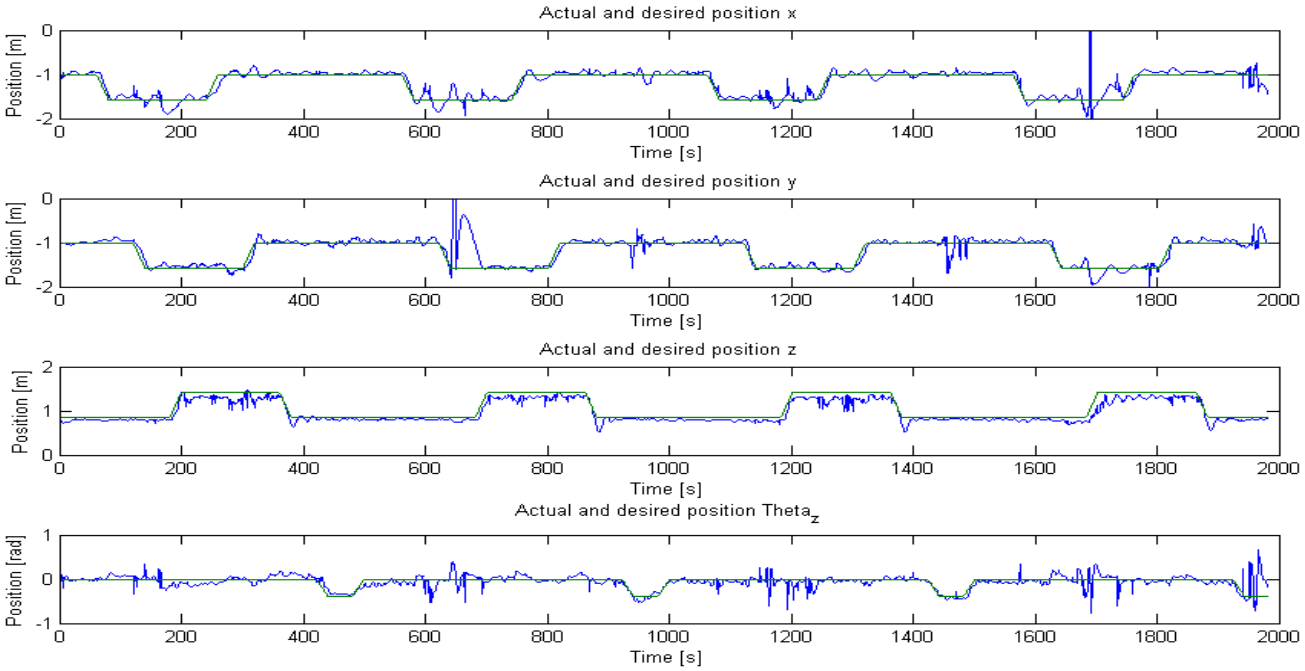

Figure 7: Test results fuzzy logic. Green: Desired position, Blue: Actual position

3. Results Sliding mode control

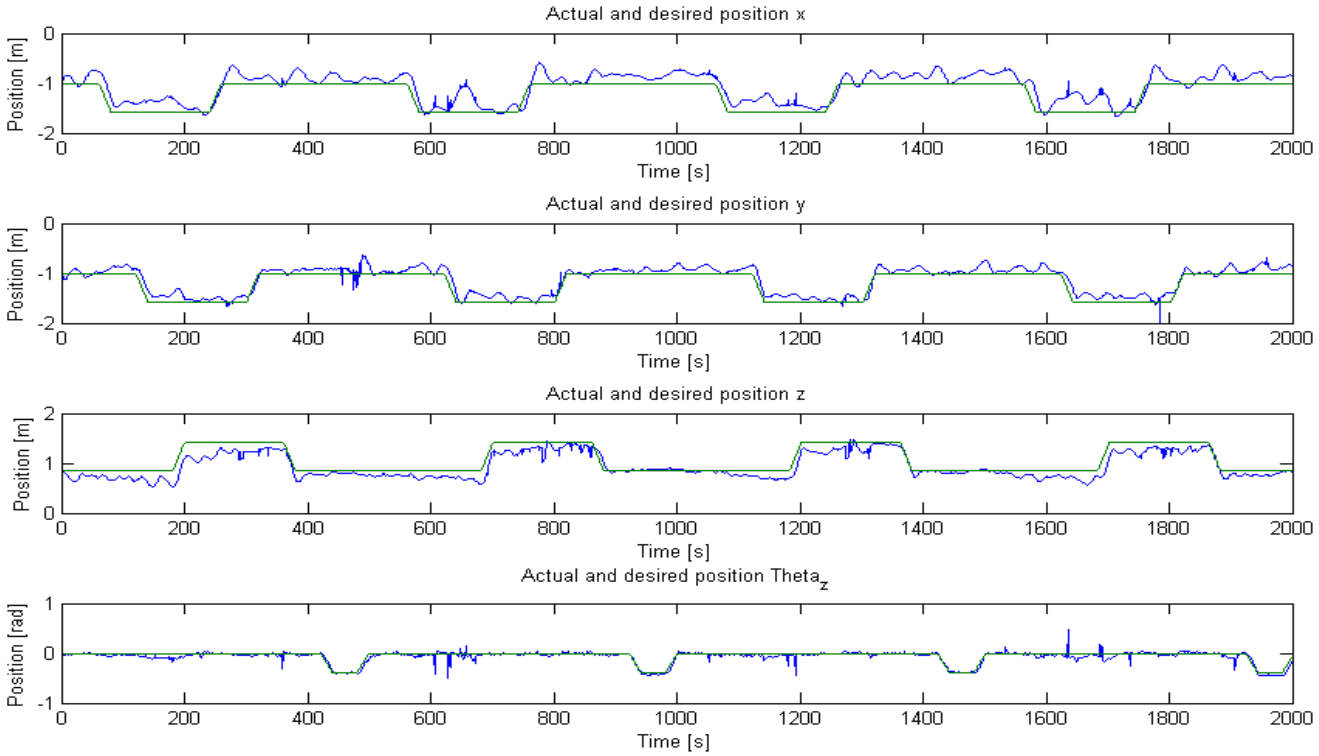

Figure 8: Test results sliding mode control. Green: Desired position, Blue: Actual position

4. Results Pre-calculated couple control

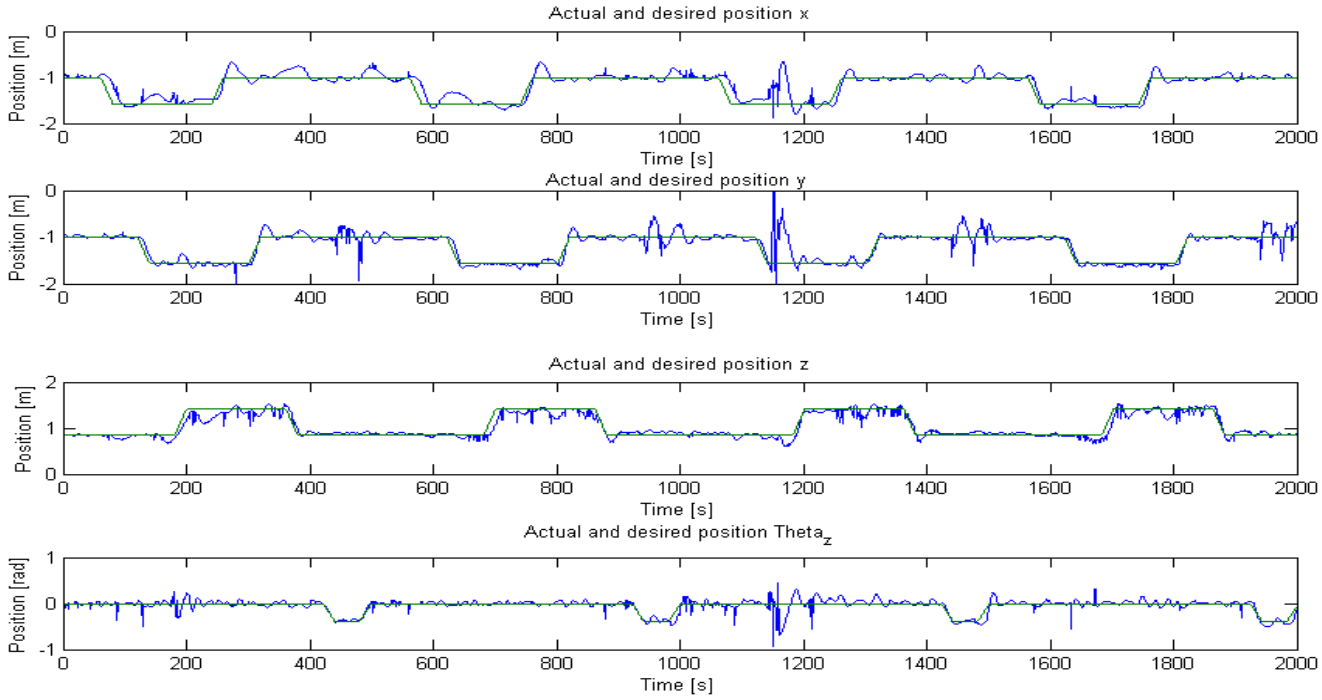

Figure 9: Test results pre-calculated couple. Green: Desired position, Blue: Actual position

D. Conclusion

The most accurate set point following is provided by the fuzzy logic controller. The downside of the fuzzy logic is that during the tests there occurred some “freak outs”. On the recording this can be seen at around 630 and 1700 seconds (see figure 7), where the blimp actually crashed into the walls (distance 0). This might be an instability caused by the integral part of the fuzzy law and could probably be reduced by further optimizing the control parameters. Luckily however, after the crash, the robot was able to re-stabilize itself without external intervention.
The pre-calculated couple controller also succeeds to follow the set point, but not as accurately as the fuzzy logic controller. Here also, a wall-crash occurred during the tests (at around 1130 seconds, see figure 9). No crash at all had the sliding mode controller. This supports the theory that this instabilities are caused by the integral part of the control law, as the sliding mode controller does not have any integration on the positioning error itself. Obviously, having no integral part on the error causes an offset in set point following, what can be seen in figure 8.

During the tests of about 30 minutes, changing battery power and environmental conditions did not seem to be an issue for any of the controllers. Because of the missing adaptive part, it was expected that the fuzzy logic would eventually fail at some point, but this did not happen. The contribution of the adaptive parts in sliding mode and pre-calculated couple control can thus not be seized with these recordings. Further tests, turning the adaptive parts on and off as well as voluntarily changing environmental conditions would have to be done.

Based on the results, the question which controller to use during performances has to be answered as follows: If accurate set point following is needed and some occasional loss of control (wall crash) is allowed, the fuzzy control implemented is the best fit. If loss of control is not acceptable at any point, the sliding mode controller has to be used, at the cost of a positioning offset.

For future work the problem of occasional loss of control has to be targeted, in order to be able to use fuzzy logic at any time. Further on, the controller will be extended on the angles around x- and y-axis in order to increase overall stability.

References

[1] Website http://www.korebot.com

[2] Reeves, N., Nembrini, J., Poncet, E., Martinoli, A. and Winfield, A., *“Voiles / Sails : Self-Assembling Intelligent Lighter than air Structures”,* Communication at the Generative Art 2005 Conference, Milan, December 2005

[3] Romano de Santis*, Class notes ELE6207, Commande des systèmes robotiques,* École Polytechnique
de Montréal, 2006