UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Computer Vision Laboratory
Institute for Systems and Robotics
VISLAB

VISION BASED CONTROL OF AN AUTONOMOUS BLIMP (VIDEOBLIMP)

Filipe Manuel da Silva Metelo, nº 46526
Luís Ricardo Garcia Campos, nº 46599

LICENCIATURA EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
Relatório de Trabalho Final de Curso
066/2002/L

Prof. Orientador: *José Santos-Victor*
Prof. Acompanhante: *Alexandre Bernardino*

Setembro de 2003

Agradecimentos

Ao longo deste trabalho, várias pessoas contribuíram com a sua valiosa ajuda e opinião. Aqui queremos deixar a nossa palavra de agradecimento.

Em primeiro lugar queremos agradecer ao nosso orientador, Prof. José Alberto Santos-Victor, que nos proporcionou a realização deste trabalho. As suas opiniões e indicações foram sempre prontas e encorajadoras.

Igualmente, queremos agradecer ao Eng. Nuno Gracias por toda a sua ajuda e insistência. A sua participação e discussões que fomos tendo foram cruciais bem como a cedência de alguns dos seus algoritmos de processamento de imagem.

Ao Eng. Alexandre Bernardino o nosso obrigado pelos inúmeros esclarecimentos em relação ao software de controlo, C++ e Windows. Agradecemos o seu interesse, simpatia e cabo USB-Serial que nos permitiu confirmar que a COM2 estava realmente queimada.

A todas as pessoas do laboratório, deixamos uma palavra de apreço pela boa disposição e convívio que nos proporcionaram.

Finalmente, obrigado a todos os nossos colegas de curso que nos acompanharam nestes longos 5 anos e fizeram com que o Técnico fosse mais que um curso de engenharia.

Eu, Filipe, quero agradecer,

A todas aquelas pessoas que estiveram presentes, nos momentos difíceis e alegres que povoaram estes últimos anos da minha vida.

Aos Engs. João Maciel e César Silva pelas sugestões e pela motivação extra.

Ao meu colega Luís. Apesar de todas as situações difíceis que foram surgindo e de todos os atrasos conseguimos levar este trabalho a bom porto.

Ao Zé Maria (Zézito), ao Sérgio, ao Filipe (Filas), Márcia e Rita (pelas férias possíveis), ao Mário e ao Manuel (apear de tudo serão sempre gajos de Electro), à Ana (sempre disposta a ajudar), ao pessoal do INESC e ao pessoal do 5º andar da torre, o meu muito obrigado.

Quero agradecer toda a paciência e dedicação dos meus pais. Eles suportaram todos aqueles dias em que passei sentado a frente do PC sem lhes ligar nenhuma e que me ouviram nos momentos mais difíceis. Afinal foram eles os principais responsáveis por eu estar aqui.

Finalmente, ao meu irmão, que mesmo sem perceber muito do que eu lhe explicava, me ouvia falar deste trabalho e de como corria e esteve sempre presente para dar a sua opinião, e à minha avó Luísa, que acabou por me ver menos vezes do que gostava.

Só queria poder, agora, ir para casa e dizer: “Estás a ver, Nanã, o teu neto já é Engenheiro!”.

Eu Luis, quero agradecer,

Em primeiro lugar ao meu colega de TFC pelo apoio, paciência e “luta” conjunta ao longo deste trabalho.

Aos meus pais não só pelo apoio demonstrado nestes meses em que o trabalho se desenvolveu mas igualmente ao longo de todos os anos do curso que agora termino.

Á minha irmã Marta pelas conversas que se serviram como tubo de escape em situações de pressão.

À minha irmã Vera pelos seus telefones a matar saudades e pela sua compreensão dos longos períodos em que não nos víamos.

Aos meus Avós pelo carinho, amizade de sempre!

Ao Filas e à Rita os sempre fieis amigos desde o primeiro ano, ao Zezito (Margem Sul Buddy), ao pessoal do 5º piso.

Por último, um agradecimento especial à Sofia pela paciência infinita (longas foram as várias horas que tiveste à minha espera) e pelo apoio dado em todas as ocasiões e pelo seu amor.

Acknowledgements

Throughout this work, several people contributed with their valuable help and opinions. Here, we would like to express our thanks to them.

First, we want to thank Prof. José Alberto Santos-Victor, who made it possible for us to do this work. His opinions and pointers were always ready and encouraging.

We would also like to thank Eng. Nuno Gracias for all his help and incentive. His participation and discussion of some ideas were crucial, as well as the rendering of some of his image processing algorithms.

To Eng. Alexandre Bernardino our thanks for the many clarifications regarding the control software, C++ and Windows. Thank you for the interest, sympathy and the USB-Serial cable that allowed us to confirm that the COM2 port was really burnt.

To all those working at the lab, we leave a word of appreciation for the good mood and fellowship that they provided.

Finally, thanks to all our course colleagues that accompanied us throughout these long 5 years and made Técnico more than just an engineering course.

I, Filipe, want to thank,

All those people that were there, both in the bad and the good times, which filled these last years in my life.

To Engs. João Maciel and César Silva for the usefull suggestions and the extra motivation.

To my partner, Luís. In spite of all those difficult situations that came up and all the delays we managed to take this Project to a good term.

To Zé Maria (Zézito), Sérgio, Filipe (Filas), Márcia and Rita (for the possible vacations), Mário and Manuel (they will always be Electro guys), to Ana (always willing to help), the INESC guys and the people working on the 5th floor of the tower, my most heart felt thanks.

I want to thank my parents for their dedication. They endured all those days I spent in front of the PC without even noticing them and heard me in the most difficult times. After all they were the ones responsible for me being here.

Finally, to my brother who, even without understanding heard me speaking of this project and how it was working out and was always there to give me his opinion and to my grandma Luisa, who ended up seeing me less often than she wanted to.

I just wish I could go home now and say: "Estás a ver, Nanã, o teu neto já é Engenheiro!".

I, Luis, want to thank,

To Filipe, my colleague and partner, for the support, patience and joint struggle showed throughout this work.

To my parents, not just for the support in these last few months, in which the work was developed, but also for the support showed in all this 5 years of the course that is ending now.

To my sister Marta for the many enlightening talks.

To my sister Vera for her phone calls and her comprehension, for the long periods in which we didn't see us.

To my grandparents for their love and friendship.

To Filas and Rita, my faithful first year friends, to Zézito (Margem Sul Buddy), to all the people in the 5º floor.

Finally a special thank you to Sofia by her endless support, infinite patience (for the long hours waiting for me) and her love.

Resumo

Para que se possa efectuar tarefas de posicionamento ou seguimento de trajectórias com um *robot* flutuante torna-se necessário o desenvolvimento de algoritmos de controlo que ultrapassem as limitações subjacentes à dinâmica e cinemática deste, além de todas as perturbações que surgem devido ao meio em que está imerso. Para mais, no caso deste trabalho, o único sensor usado é um sistema de visão que consiste numa micro câmara colocada a bordo do *robot* e cujas imagens são sujeitas a processamento em tempo-real. Das homografias entre imagens consecutivas e assumindo alguns pressupostos sobre o ambiente em redor, torna-se possível estimar velocidades e deslocamentos do *robot* no espaço 3D.

Neste trabalho desenvolvem-se métodologias de controlo que permitem o cumprimento das tarefas de posicionamento ou seguimento de trajectórias, ultrapassando as limitações impostas, quer pelo sistema físico em sí, quer pelo sensor. São apresentados e estudados algoritmos de processamento de imagem que permitem obter a pose e a velocidade do veículo. Abordam-se vários tipos de controlo linear e não linear de modo a fazer o controlo do valor da velocidade do veículo e sua direcção no espaço 3D. Duas estratégias de definição das referências são propostas, uma baseada em posições e coordenadas no espaço Cartesiano e outra baseada em medições feitas nas imagens da câmara, permitindo assim reduzir os efeitos de eventuais erros na calibração da câmara.

O trabalho passa, numa primeira fase, pela modelação do sistema e identificação de parâmetros e teste do controlo e processamento de imagem num simulador desenvolvido para esse efeito. Seguidamente são feitas experiências num *setup* real em que se implementam os vários algoritmos em tempo-real.

Palavras-chave: Dirigível, Identificação de Parâmetros, Visão, Homografias, Controlo Linear e Não-Linear, Controlo em Tempo-Real, Seguimento de Trajectórias.

Abstract

In order for it to be possible to perform positioning or trajectory following tasks with a floating robot the development of control algorithms that overcome the underlying limitations of the system's dynamics and kinematics, as well as the external disturbances, is required. In the case this project, the only sensor used is a vision system consisting of a micro camera placed onboard the robot whose images are subject to real-time processing. From the homographies between consecutive images and assuming some priors regarding the surrounding environment, it is possible to estimate velocities and displacements of the robot in 3D space.

In this work, we develop control methodologies that enable the system to accomplish positioning or trajectory following tasks, surpassing some limitations imposed by the physical system and the sensor. Image processing algorithms that enable us to obtain the vehicle's pose and velocity are presented and studied. Several types of linear and non-linear control are approached in order to control the velocity value of the vehicle as well as its heading in 3D space. Two strategies for the reference definition are proposed, one based in position and coordinates in Cartesian space and the other based in image measurements, thus avoiding the need for high precision camera calibration.

The work developed consists, firstly, in the modelling and parameter identification of the system and control and image processing tests in a specially developed simulator. Secondly, experiments are made with the real setup in which the algorithms are implemented, running in real-time.

Key-words: Blimp, Parameter Identification, Vision, Homographies, Linear and Non-Linear Control, Real-Time Control, Trajectory Tracking.

Table of Contents

List of Figures

List of Tables

1. Introduction

In this chapter, we present the motivation for this work and provide some of the necessary references to other related work. Some important contributions are presented as well as an overall overview.

1.1. Motivation

Autonomous aeronautical vehicles have recently gained importance, not only as the subject of research but, also as a useful means of advertising, climate research, surveillance or even infiltration in war scenarios. A common denominator in all these situations is the impossibility or undesirability of having human presence at the scene.

A blimp is an aerial vehicle that has better performance than small helicopters or airplanes in applications that require low speed and altitude and is less noisy, which can be a great advantage in the tasks referred. Also very important is the fact that this vehicle is inherently stable and this can lead to more simple control mechanisms and assurance that the vehicle in not destroyed or mangled in the event of a system failure. Furthermore, there has been recent investigation on the role of this type of craft in the moving of heavy. Big enough blimps can facilitate the transport of big and heavy cargo that would otherwise have to be moved in many different parts by regular transportation such as train, truck or boat.

The capacity to maintain constant velocity or pose is an essential requisite to perform any navigation task, which is desired for the vehicle. In order to fulfil this requirement, it primarily becomes necessary to measure the robot's velocities and pose with an accurate sensor. The existence of some control and navigation algorithm is essential to ensure reference following and, thus, the desired behaviour.

In unstructured environments where it is not possible to obtain absolute position estimates like the ones given by GPS (Global Positioning System), vision is a powerful sensor for controlling the system. The use of vision in the loop for autonomous floating vehicles is an important research area at ISR (Institute for Systems and Robotics), especially at the Computer Vision Laboratory, and this project aims at contributing to the related research performed there.

1.2. Previous Work

As we have referred, there has been a series of projects at ISR, more specifically at the Computer Vision Lab (where this work was developed), related to the field of visual control of floating robots. Past projects include NARVAL (Navigation of Autonomous Robots via Active Environmental Perception) and ranged from such diverse areas as mosaic building [1], visual servoing based on built mosaics [1, 2, 3] and visual station keeping [2]. The methodologies and algorithms developed in these projects were applied in the control loop of an underwater ROV (Remote Operated Vehicle) and an indoor blimp. This past work focused mainly on vision processing algorithms for motion estimation. As the setups used are naturally stable, the dynamics were neglected, assuming purely kinematical systems for the algorithm development, using manually tuned PID (Proportional Integral Derivative) controllers. These algorithms assume that the world is almost planar and is represented by a mosaic image built previously, and therefore known to the system prior to the experiments.

Other teams have developed theory for vision based control of robotic manipulators, eventually demonstrating the implications of different methodologies in the robustness of the closed loop system [4]. Although considerations developed around the system dynamics are not very useful in the control of autonomous floating vehicles, the hierarchies presented and sum of the possible vision configurations in [4] are perfectly applicable to the case in study.

As for floating vehicles dynamics, control and modelling, this is extensively studied in literature [5]. The referred book presents the analytical tools for the complete modelling of any of these vehicles and presents possible control solutions for the problems in hand.

1.3. Objectives and Contributions

The work developed is connected, mainly, to the development, implementation and testing of control and navigation algorithms of a floating robot whose only sensor is an onboard black and white, wireless micro camera.

The image matching algorithms, developed in [1], are used in the vision processing routines. These algorithms can provide us absolute pose and odometry, with some limitations.

The dynamics model of an aerial vehicle is presented and discussed, along with model simplifications and related problems. System identification is also necessary and we present here the necessary steps in order to obtain a realistic model of the real airship.

Both linear control and non-linear robust control are studied and the results are compared. A simple trajectory following behaviour is also presented and tested under several different constraints. Note that the image processing algorithms are slow and we are in the presence of a very noisy sensor with some calibration related bias, and the controller is required to overcome all these serious difficulties.

Algorithms for smooth trajectory planning and path following are implemented and adapted for this case, where the robot has some lateral but uncontrolled drift, caused by actuation. This drift makes the path following problem harder than in most common situations found in the literature, where there is no drift, and where trajectory planning is well developed for non-holonomic vehicles [6,7].

The fulfilment of our objectives also includes the implementation of a simulator to accelerate the development of the algorithms and allow for a more in depth analysis of the results. This simulator is also required to be modular and its internal functioning easy to learn, as it will be a working tool for other people at the lab to test their algorithms.

1.4. Structure of the Report

This report is organized as follows. Chapter 2 regards the theoretical study of the robot’s physics. Here we present the dynamics and kinematics Equations for the blimp and actuators (propellers). Chapter 3 addresses sensor modelling and, therefore, presents the geometrical description of image formation. The image processing algorithms are also studied and we present the way to obtain the system’s state based on the results of this processing. In Chapter 4, we focus on the controller design and in Chapter 5 trajectory following algorithms are developed. In Chapter 6, experiments, both in the simulation environment and with the real setup, are documented and conclusions are drawn. Finally, Chapter 7 presents a perspective over the results of this work and possible future contributions are suggested.

2. Robot Modelling

In this chapter, we present a detailed analysis of the physics of the system studied in this work. First, a short description of the real vehicle is done. Second, the non-linear model of the blimp, as well as the actuators, is presented and simplifications to the models are discussed. In the last part of this chapter, we use linearization to further simplify the system's model and some comments are done on the validity of the simplified model.

2.1. The Blimp and Actuators

The test bed vehicle is a simple radio controlled indoor blimp. It is composed by a large ellipsoidal mylar bag filled with helium, for lift, with four tail fins for movement stabilisation and a gondola. The gondola, attached to the bottom of the bag, contains the radio receiver, motors drive, servo and two lateral thrusters. The stern thruster is placed in the lower tail fin. The location of the thrusters is shown Figure 2.1.

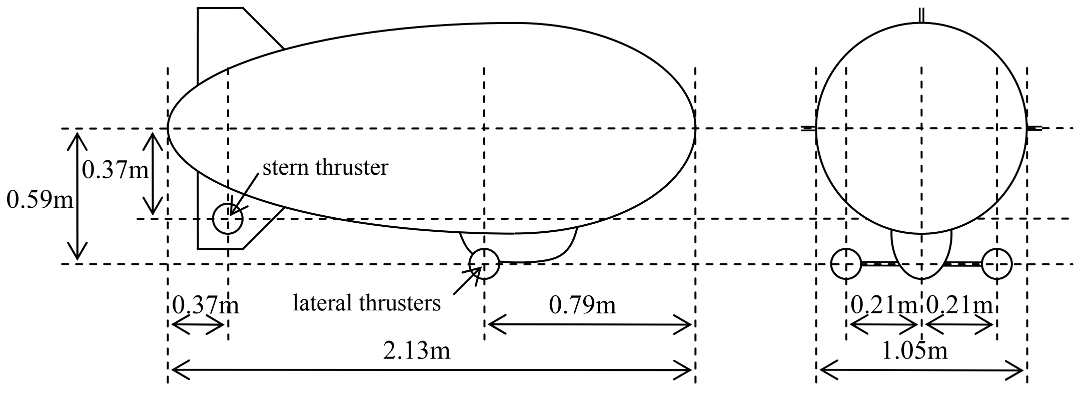


Figure 2.1 - Location of the thrusters

As shown in Figure 2.2, it is in the gondola that the servo and axis that control the lateral thrusters' direction are located. The lateral thrusters (starboard and portside) control vertical plane displacement and the stern thruster controls heading.

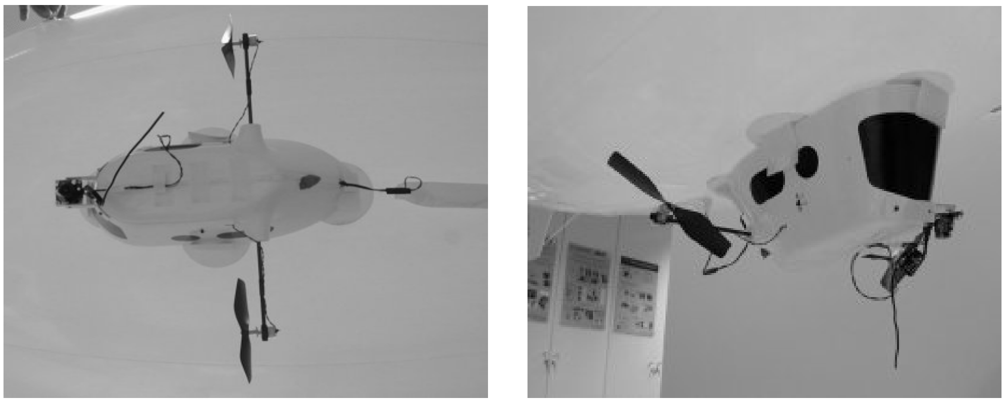


Figure 2.2 - The blimp's gondola

2.2.The Non-linear Model of the Blimp

In the next sections, the conventions used in the kinematics equations and the dynamics' equations that describe the system's behaviour are presented.

2.2.1. Kinematics

The kinematics description of the vehicle is based in the existence of two frames, one placed in the blimp's body, at the buoyancy centre (the blimp's frame $\{b\}$), and the other in the ground plane, (the world frame $\{w\}$). The ground is assumed to be locally planar and we will use $\{w\}$ as the inertial frame (neglecting the rotation movement of the Earth).

The x_b axis (from $\{b\}$) points towards the front of the vehicle, i.e. in the prow direction and the z_b axes is placed pointing down towards the floor, in the vertical direction. The y_b axis, in order for it to be orthogonal to the other axes, points right if you are looking forward, i.e. points to starboard. Note that $\{w\}$ has a similar configuration with z_w pointing down.

The frames that were just described are presented in Figure 2.3.

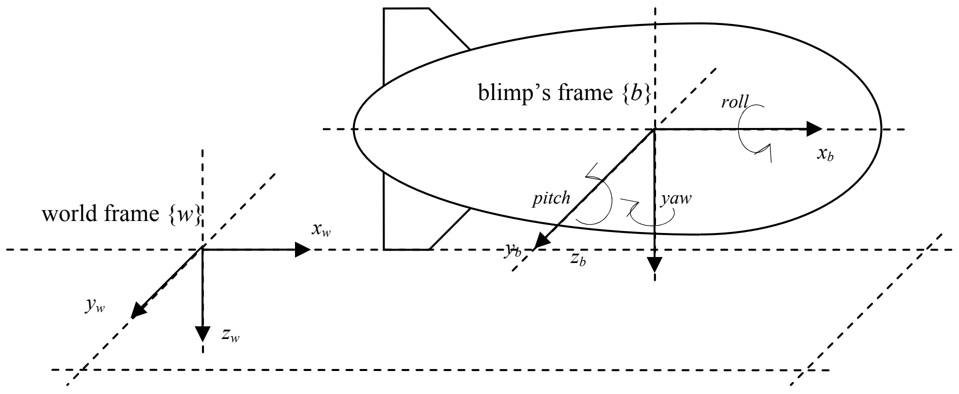


Figure 2.3 - Placement of the reference frames

Therefore, in the case of any vehicle moving in 3D space, the physical variables that are used to describe the kinematics model of the system are:

$$\begin{aligned}
\eta &= \left[\eta_1^T, \eta_2^T\right]; \quad & \eta_1^T &= \left[x, y, z\right]^T; \quad & \eta_2^T &= \left[\phi, \theta, \psi\right]^T; \\
v &= \left[v_1^T, v_2^T\right]; \quad & v_1^T &= \left[v_x, v_y, v_z\right]^T; \quad & v_2^T &= \left[w_x, w_y, w_z\right]^T; \\
\tau &= \left[\tau_1^T, \tau_2^T\right]; \quad & \tau_1^T &= \left[F_x, F_y, F_z\right]^T; \quad & \tau_2^T &= \left[N_x, N_y, N_z\right]^T;
\end{aligned} \tag{2.1}$$

The vector η contains the coordinates of the blimp's frame $\{b\}$ (the blimp's pose) in $\{w\}$ while the v and τ vectors represent the velocities and applied forces, described in $\{b\}$.

In order to have the kinematics equations for the vehicle it is necessary to devise a formula that converts the variables from the blimp's frame to the world frame. The jacobian that describes this transformation is presented below.

$$\dot{\eta}_1 = J_1(\eta_2)v_1; \quad J_1(\eta_2) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\theta + c_\psi s_\theta s_\phi & s_\psi s_\theta + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\theta + s_\psi s_\theta s_\phi & -c_\psi s_\theta + s_\theta s_\psi c_\phi \\ -s_\theta & c_\psi s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.2}$$

$$\dot{\eta}_2 = J_2(\eta_2)v_2; \quad J_2^{-1}(\eta_2) = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix}; \quad J_2(\eta_2) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \tag{2.3}$$

$$\begin{bmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \end{bmatrix} = \begin{bmatrix} J_1(\eta_2) & [0]_{3\times 3} \\ [0]_{3\times 3} & J_2(\eta_2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Leftrightarrow \dot{\eta} = J(\eta)v \tag{2.4}$$

Equation (2.4) allows us to transform vectors from one frame to the other and vice-versa. This model is used for the vehicle's kinematics.

2.2.2. Dynamics

The blimp's dynamics can be conveniently written in the form of the following equation, whose variables are described in frame $\{b\}$:

$$M\dot{v}_b + C(v_b)v_b + D(v_b)v_b + g(\eta_b) = \tau_b \tag{2.5}$$

$M = M_{RB}+M_A =$	mass matrix (including added mass terms)
$C(v_b) = C_{RB}(v_b)+C_A(v_b) =$	Coriolis matrix and centripetal terms (including added mass terms)
$D(v_b) =$	hydrodynamic *damping*
$g(\eta_b) =$	restoring forces vector (from gravity and buoyancy)
$\tau_b =$	disturbance and actuation forces and torques

The matrixes that model the blimp's dynamics in relation to $\{b\}$ are described in the next paragraphs. Simplifications made come, especially, from not taking into account the gondola (where the thrusters are placed) and the deflectors, thus approximating the shape of the blimp by the shape of the envelope. These are perfectly valid if the vehicle is moving at low speed and the size of the gondola is small in comparison with the envelope, which is the case here. Moreover, the shape of the envelope is assumed to have some symmetry.

Assuming that the blimp is an ellipsoid with major semi-axis a, minor semi-axis b and origin corresponding to the origin of $\{b\}$ we have, for the inertia coefficients matrix:

$$I_b = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \cong \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.6}$$

Assuming the vehicle to be non-deformable, we can obtain the Equations that describe the rigid body mass matrix M_{RB}. The $[x_G\ y_G\ z_G]$ vector represents the location of the centre of mass with relation to the origin of the body frame $\{b\}$. As the gondola is placed under the envelope, below the blimp's frame $\{b\}$, we assume that it is possible to distribute the weight in order for the centre of mass to have null y_G.

$$M_{RB} = \begin{bmatrix} m[I]_{3\times 3} & -mS \\ mS & I_b \end{bmatrix}, \quad S = \begin{bmatrix} 0 & -z_G & y_G \\ z_G & 0 & -x_G \\ -y_G & x_G & 0 \end{bmatrix} \tag{2.7}$$

Any vehicle immersed in fluid, in order to move, needs to displace some quantity of fluid and therefore there will be induced forces and moments, which can be understood as if its mass is greater than it really is. This is why there is the necessity to incorporate some added mass terms in the already presented mass matrix, in order to describe this phenomenon and obtain the full mass matrix to use in the vehicle's model.

As we have pointed out before, the blimp moves at low speed and hase symmetries in several axes so we can consider the simplified added mass matrix M_A presented.

$$M_A = diag\{a_{11}, a_{22}, a_{33}, a_{44}, a_{55}, a_{66}\} \tag{2.8}$$

It can be proved [1] that, in the presence of an ideal fluid, the added mass matrix is strictly positive definite and thus the full mass matrix has the form presented below.

$$M = \begin{bmatrix} m+a_{11} & 0 & 0 & 0 & mz_G & 0 \\ 0 & m+a_{22} & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m+a_{33} & 0 & -mx_G & 0 \\ 0 & -mz_G & 0 & I_{xx}+a_{44} & 0 & 0 \\ mz_G & 0 & -mx_G & 0 & I_{yy}+a_{55} & 0 \\ 0 & mx_G & 0 & 0 & 0 & I_{zz}+a_{66} \end{bmatrix} \tag{2.9}$$

Next, we will present the Coriolis and centripetal terms' matrix. This matrix has terms related both to the rigid body mass and to the added mass terms. Again, assuming the location of the centre of mass to have $y_G=0$, the simplified matrix for the rigid body is:

$$C_{RB}(v_b) = \begin{bmatrix} 0 & 0 & 0 & mz_G w_z & -m(x_G w_y - v_z) & -m(x_G w_z + v_y) \\ 0 & 0 & 0 & -mv_z & m(z_G w_z + x_G w_x) & mv_x \\ 0 & 0 & 0 & -m(z_G w_x - v_y) & -m(z_G w_y + v_x) & mx_G w_x \\ -mz_G w_z & mv_z & m(z_G w_x - v_y) & 0 & I_{zz} w_z & -I_{yy} w_y \\ m(x_G w_y - v_z) & -m(z_G w_z + x_G w_x) & m(z_G w_y + v_x) & -I_{zz} w_z & 0 & I_{xx} w_x \\ m(x_G w_z + v_y) & -mv_x & -m(x_G w_x) & I_{yy} w_y & -I_{xx} w_x & 0 \end{bmatrix} \tag{2.10}$$

As for the matrix resulting from the added mass terms, we can obtain a simplified version of it by overlooking the non-linear components (which are very small in the case of a slow moving vehicle) and thus obtaining the added mass terms Coriolis matrix and the resulting full Coriolis and centripetal terms' matrix:

$$C_A(v_b) = \begin{bmatrix} 0 & 0 & 0 & 0 & a_{33} v_z & -a_{22} v_y \\ 0 & 0 & 0 & -a_{33} v_z & 0 & a_{11} v_x \\ 0 & 0 & 0 & a_{22} v_y & -a_{11} v_x & 0 \\ 0 & a_{33} v_z & -a_{22} v_y & 0 & a_{66} w_z & -a_{55} w_y \\ -a_{33} v_z & 0 & a_{11} v_x & -a_{66} w_z & 0 & a_{44} w_x \\ a_{22} v_y & -a_{11} v_x & 0 & a_{55} w_y & -a_{44} w_x & 0 \end{bmatrix} \tag{2.11}$$

$$C(v_b) = \begin{bmatrix} 0 & 0 & 0 & mz_G w_z & -mx_G w_y + (m+a_{33}) v_z & -mx_G w_z - (m+a_{22}) v_y \\ 0 & 0 & 0 & -(m+a_{33}) v_z & m(z_G w_z + x_G w_x) & (m+a_{11}) v_x \\ 0 & 0 & 0 & -mz_G w_x + (m+a_{22}) v_y & -mz_G w_y - (m+a_{11}) v_x & mx_G w_x \\ -mz_G w_z & (m+a_{33}) v_z & mz_G w_x - (m+a_{22}) v_y & 0 & (I_{zz}+a_{66}) w_z & -(I_{yy}+a_{55}) w_y \\ mx_G w_y - (m+a_{33}) v_z & -m(z_G w_z + x_G w_x) & mz_G w_y + (m+a_{11}) v_x & -(I_{zz}+a_{66}) w_z & 0 & (I_{xx}+a_{44}) w_x \\ mx_G w_z + (m+a_{22}) v_y & -(m+a_{11}) v_x & -m(x_G w_x) & (I_{yy}+a_{55}) w_y & -(I_{xx}+a_{44}) w_x & 0 \end{bmatrix} \tag{2.12}$$

The aerodynamic damping matrix comes, mainly, from the modelling of the skin friction of the vehicle with the fluid it is immersed in. Again, assuming the previously explained simplifications, we conclude that third and higher order terms can be neglected and so the matrix will have a simple form.

$$D=diag\{D_{v_x}+D_{v_xv_x}|v_x|,D_{v_y}+D_{v_yv_y}|v_y|,D_{v_z}+D_{v_zv_z}|v_z|,D_{w_x}+D_{w_xw_x}|w_x|,D_{w_y}+D_{w_yw_y}|w_y|,D_{w_z}+D_{w_zw_z}|w_z|\} \quad (2.13)$$

Finally, we are left with the description of the restoring forces vector, which expresses the influence of the gravity and buoyancy forces in the dynamics' behaviour. Assuming that the vehicle is neutrally buoyant, i.e. the gravity force is equal, in value (opposite in direction), to the buoyancy force and using $y_G=0$ the restoring forces vector comes simplified to:

$$g(\eta_b)=\begin{bmatrix} [0]_{3\times 1} \\ -mg\cos(\theta)\sin(\phi)y_G+mg\cos(\theta)\sin(\phi)z_G \\ mg\sin(\theta)z_G+mg\cos(\theta)\cos(\phi)x_G \\ -mg\cos(\theta)\sin(\phi)x_G-mg\sin(\theta)y_G \end{bmatrix} \equiv \begin{bmatrix} [0]_{3\times 1} \\ mg\cos(\theta)\sin(\phi)z_G \\ mg\sin(\theta)z_G+mg\cos(\theta)\cos(\phi)x_G \\ -mg\cos(\theta)\sin(\phi)x_G \end{bmatrix} \quad (2.14)$$

In Appendix A, the parameter identification experiments made with the blimp are systematized and numeric values for the model here described are calculated and presented.

2.3. The Actuators' Theoretical Model

The modelling of the actuators used is presented in the next sections. Both the nonlinear kinematics and the simplified dynamics are presented with justification for the models used.

2.3.1. Kinematics

The actuators used are small DC motors equipped with propellers and therefore they will have a non-linear gain characteristic. The model adopted to describe these thrusters is the bilinear model [5]. This was chosen because it provides an adequate, yet simple representation of the non-linearities from the aerodynamic damping in the propellers.

The forces developed by the thrusters can be modelled by the following equation:

$$F=\rho D^4 K_T\left(\frac{Va}{nD}\right)|n|n \quad (2.15)$$

In this equation, n represents the rotation speed, D the diameter of the propeller, ρ the fluid density and K_T the thrust coefficient that is dependent on n, D and V_a, the velocity at which the fluid passes through the propeller when the vehicle is moving.

The behaviour of this function is, in general, diverse whether the rotation is in the positive or negative direction, due to the propeller not being symmetrical. In each case, K_T can be approximated by a linear function with V_a depending on the speed of the vehicle and w (wake fraction), which typically presents values of 0.1 to 0.4.

$$K_T=\alpha_1+\alpha_2\frac{Va}{nD}, \quad Va=(1-w)V \quad (2.16)$$

The final result for the force exerted by the thrusters as a function of rotation speed and velocity of the vehicle is given by the following expression. The relation between the speed of rotation *n* and the control value applied is described in the next Section, referring to the actuator dynamics.

$$F = b_1 |n| n + b_2 |n| v \tag{2.17}$$

Depending on the location of the thrusters in the vehicle's body frame $\{b\}$ there will be a relation that converts locally applied forces into resulting forces and moments described in $\{b\}$. The configuration of the thrusters is schematically represented in Figure 2.3, below, and directions of the applied forces are defined.

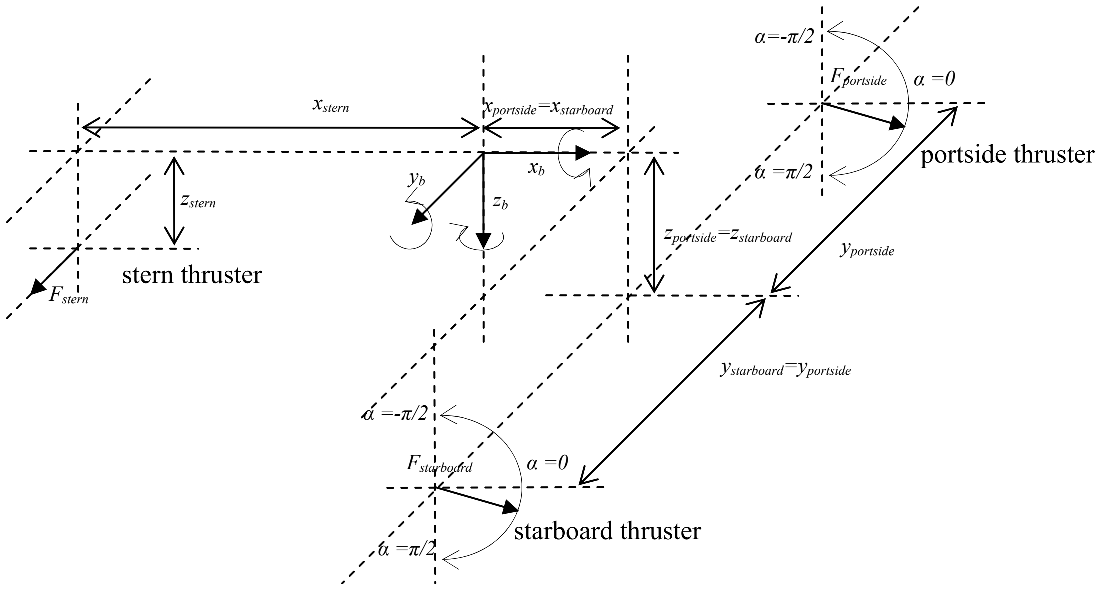


Figure 2.4 - Placement of the thrusters in the body fixed frame $\{b\}$.

Therefore, the force vector $\tau_b=[F_x\ F_y\ F_z\ N_x\ N_y\ N_z]^T$, defined in the previous section, can be obtained, as described in [8], by the following expressions (assuming symmetry between the two lateral thrusters).

$$\begin{aligned} F_x &= (F_{portside} + F_{starboard})\cos(\alpha) \\ F_y &= F_{stern} \\ F_z &= (F_{portside} + F_{starboard})\sin(\alpha) \end{aligned} \tag{2.18}$$

$$\begin{aligned} N_x &= -F_y z_{stern} \\ N_y &= -F_z x_{portside} + F_x z_{portside} \\ N_z &= F_y x_{stern} \end{aligned} \tag{2.19}$$

The precise measurement of the placement of the thrusters and the experimental tests that lead to the non-linear gain function are presented in Appendix A. Considerations about the model just presented are made and the final charts with the description of these non-linear functions are presented.

2.3.2. Dynamics

The propellers used are coupled to simple DC motors, two of which (the lateral thrusters) are coupled to the servo that controls the angle α.

In general, the model that represents a small DC motor, with a propeller, immersed in a fluid is described by the equations below:

$$\begin{aligned}&T = k_m i_a\\&J\dot{n} = T - k_{lin}n - k_{sqr}n|n| \Leftrightarrow T = J\dot{n} + k_{lin}n + k_{sqr}n|n|\\&V = i_a R_a + L_a \dot{i}_a + k_b n \cong i_a R_a + k_b n\end{aligned} \qquad (2.20)$$

V =	induced voltage [V]
i_a =	induced current [A]
n =	angular velocity of the rotor [rad]
J =	inertia of the rotor and load [kgm^2]
k_{lin} =	viscous damping coefficient for the rotor's supports [Nm/rad/s]
k_{sqr} =	non-linear damping coefficient from fluid flowing through the propeller [Nm/rad^2/s^2]
R_a =	induced circuit's resistance [Ω]
L_a =	induced circuit's impedance [H]

Assuming that the model is approximately first order, we can describe the system as linear with one stable pole (if the output is velocity). In reality this is not quite so, as the place for the pole of the system depends on the load and the rotation velocity of the propeller. Anyway, as the pole's value is not greatly changed and is, in any case, much faster than the blimp's faster poles, we assume that it is at a constant value.

The static gain value for the motor is modelled as being unitary, and all the non-linear effects described here can be included in the determination of the gain non-linearities made experimentally, as referred in the previous Section.

Thus, the model chosen to represent the dynamics of the thrusters is shown below (*n* is the speed of rotation of the propeller and *V* is the applied voltage value):

$$\frac{n(s)}{V(s)} = \frac{a}{s+a} \qquad (2.21)$$

Saturation in the drive of the motors, as well as dead zone, is not addressed in the model described here, but it exists and must be contemplated in the definition of the actuation values.

2.4. Analysis and Simplification of the Models

As expressed in Equation (2.5), the vehicle in study can be modelled by a predominantly non-linear set of equations of reasonable complexity. Therefore, in order to perform the controller design for this system it becomes necessary to incur in several simplifications. This facilitates the analysis of the system, the use of known control methodologies and the choice of state variables to control with each available actuator.

One of the main characteristics of this type of airship is its inherent stability, as shown in [5], and therefore it is not necessary for the control to act in stabilization, so good performance in reference following will be our main objective.

In order to obtain the linearized model of the blimp it is necessary to define equilibrium values either for velocity or for pose. These are defined as:

$$\begin{aligned} \eta_0(t) &= [x_0(t), y_0(t), z_0(t), \phi_0(t), \theta_0(t), \psi_0(t)] \\ v_0(t) &= [v_{x0}(t), v_{y0}(t), v_{z0}(t), w_{x0}(t), w_{y0}(t), w_{z0}(t)] \end{aligned} \tag{2.22}$$

Disturbance in the vicinity of these values is represented as:

$$\Delta\eta(t) = \eta(t) - \eta_0(t); \quad \Delta v(t) = v(t) - v_0(t); \tag{2.23}$$

We can therefore linearize the Newton-Euler equation presented in (2.5):

$$\mathrm{M}\Delta\dot{v} + \left.\frac{\partial C(\mathrm{v})\mathrm{v}}{\partial}\right|_{v_0} \Delta v + \left.\frac{\partial D(\mathrm{v})\mathrm{v}}{\partial}\right|_{v_0} \Delta v + \left.\frac{\partial \mathrm{g}(\eta)}{\partial}\right|_{\eta_0} \Delta\eta = \tau \tag{2.24}$$

As for the kinematics equation we can apply the disturbances and, by overlooking the second order terms, we have:

$$\begin{aligned} \dot{\eta}_0 + \Delta\dot{\eta} &= J(\eta_0 + \Delta\eta)(v_0 + \Delta v) \Leftrightarrow \\ \Delta\dot{\eta} &= J(\eta_0 + \Delta\eta)(v_0 + \Delta v) - J(\eta_0)v_0 \Leftrightarrow \\ \Delta\dot{\eta} &= J(\eta_0 + \Delta\eta)\Delta v - \left(J(\eta_0 + \Delta\eta) - J(\eta_0)\right)v_0 \Leftrightarrow \\ \Delta\dot{\eta} &\approx J(\eta_0)\Delta v - \left(J(\eta_0 + \Delta\eta) - J(\eta_0)\right)v_0 \end{aligned} \tag{2.25}$$

Defining $x_1 = \Delta v$, $x_2 = \Delta\eta$ and $Bu = \tau$, the following time invariant model is obtained:

$$\begin{aligned} M\dot{x}_1 + C(t)x_1 + D(t)x_1 + G(t)x_2 &= Bu \\ \dot{x}_2 &= J(t)x_1 + J^*(t)x_2 \end{aligned} \tag{2.26}$$

$$\text{with} \begin{cases} C(t) = \left.\dfrac{\partial C(v)v}{\partial v}\right|_{v_0(t)}; D(t) = \left.\dfrac{\partial D(v)v}{\partial v}\right|_{v_0(t)}; G(t) = \left.\dfrac{\partial g(\eta)}{\partial \eta}\right|_{\eta_0(t)} \\ J(t) = J(\eta_0(t)); J^*(t) = \left(J(\eta_0(t) + \Delta\eta(t)) - J(\eta_0(t))\right) \end{cases}$$

Finally, calculating the various matrixes, the linearized state-space model is:

$$\begin{bmatrix} M\dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\left(C(t) + D(t)\right) & -G(t) \\ J(t) & J^*(t) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u \tag{2.27}$$

Therefore:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -M^{-1}\left(C(t) + D(t)\right) & -M^{-1}G(t) \\ J(t) & J^*(t) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} M^{-1}B \\ 0 \end{bmatrix} u \Leftrightarrow \dot{x} = Ax + B'u \tag{2.28}$$

We assume that the steady state velocities are null, as we are looking for a model that describes a slow moving vehicle. Thus the Coriolis matrix is also null ($C(v)v=0$). Furthermore it is acceptable that, in steady state, the *roll* (ϕ) and *pitch* (θ) angles are approximately zero.

Using the assumptions made above, the matrixes for the linearized system are:

$$M = \begin{bmatrix} m+a_{11} & 0 & 0 & 0 & mz_G & 0 \\ 0 & m+a_{22} & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m+a_{33} & 0 & -mx_G & 0 \\ 0 & -mz_G & 0 & I_{xx}+a_{44} & 0 & 0 \\ mz_G & 0 & -mx_G & 0 & I_{yy}+a_{55} & 0 \\ 0 & mx_G & 0 & 0 & 0 & I_{zz}+a_{66} \end{bmatrix}$$

$$D = diag\{ D_{v_x}, D_{v_y}, D_{v_z}, D_{w_x}, D_{w_y}, D_{w_z} \}$$

$$G = \begin{bmatrix} & [0]_{3\times 6} & & \\ & z_G mg & 0 & 0 \\ [0]_{3\times 3} & 0 & z_G mg & 0 \\ & -x_G mg & 0 & 0 \end{bmatrix} \tag{2.29}$$

$$J = \begin{bmatrix} cos(\psi_0) & -sin(\psi_0) & \\ sin(\psi_0) & cos(\psi_0) & [0]_{2\times 4} \\ \multicolumn{2}{c}{[0]_{4\times 2}} & [I]_{4\times 4} \end{bmatrix}$$

By observing the model obtained with care, it is clear that this can be separated into two fully independent systems just by the switching of some lines and columns in the matrixes, as shown in the next sections. We are, therefore, in the presence of two systems, one that describes the blimp's behaviour in the vertical plane (XZ system) and the other that models the rotational behaviour in turn of the *z* and *x* axes and displacement in *y* (Heading system).

Notice that the *J* matrix is only time invariant if we consider the steady state *yaw* angle to be null, which is not true, in general.

Finally, it can be seen that the real coupling between these two systems in the full non-linear model is the result of the inherent structure of the Coriolis matrix. The effects of this, at low speed are reduced and, thus, *C(v)* can be neglected even on the non-linear model, again leaving us with the two decoupled systems. We can include in these the full non-linear dynamics from the gravitational effects and damping matrix *D(v)*. In this situation, we will be incurring in errors, as the coupling exists. However, this coupling can be considered as disturbance that the controllers have to deal with.

2.4.1. Linearized Heading System

The perturbed state variables for the heading system are $x(t)=[v_y(t)\ w_x(t)\ w_z(t)\ y(t)\ \phi(t)\ \psi(t)]^T$, and the actuation will be performed only by the stern thrusters so $u=F_{stern}$ The system is described by the state space model in Equation (2.30).

$$\dot{x} = \begin{bmatrix} -M^{-1}D & -M^{-1}G \\ J & [0]_{3\times 3} \end{bmatrix} x + \begin{bmatrix} M^{-1}B \\ [0]_{3\times 1} \end{bmatrix} u \tag{2.30}$$

The *M*, *D*, *G* and *J* matrixes are built from the elements of the ones presented in (2.29) that correspond to the chosen state variables for this system, as we show in the following equations. The *B* matrix depends on the location of the stern thrusters in $\{b\}$.

$$M = \begin{bmatrix} m + a_{22} & -mz_G & mx_G \\ -mz_G & I_{xx} + a_{44} & 0 \\ mx_G & 0 & I_{zz} + a_{66} \end{bmatrix}$$

$$D = diag\{ D_{v_y}, D_{w_x}, D_{w_z} \}$$

$$G = \begin{bmatrix} 0 & 0 & 0 \\ 0 & z_G mg & 0 \\ 0 & -x_G mg & 0 \end{bmatrix} \tag{2.31}$$

$$\mathrm{J} = \begin{bmatrix} \sin(\psi_0) & \cos(\psi_0) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ z_{stern} \\ x_{stern} \end{bmatrix}$$

2.4.2. Linearized XZ System

In this case, the perturbed state variables are $x(t)=[v_x(t)\ v_z(t)\ w_y(t)\ x(t)\ z(t)\ \theta(t)]^T$, and the actuation is $u=[F_x\ F_z]^T$. Note that F_x and F_z are considered decoupled for simplification although, in fact, this is not really the case as these two forces depend both on the orientation α of the thrusters and the one-dimensional actuation force. In reality, we have:

$$\begin{matrix} F_x = F\cos(\alpha) \\ F_z = F\sin(\alpha) \end{matrix}\ ,\quad F_{portside} = F_{starboard} = \frac{F}{2} \tag{2.32}$$

The system can be described by the following state space model:

$$\dot{x} = \begin{bmatrix} -M^{-1}D & -M^{-1}G \\ J & [0]_{3\times 3} \end{bmatrix} x + \begin{bmatrix} M^{-1}B \\ [0]_{3\times 2} \end{bmatrix} u \tag{2.33}$$

The *M*, *D*, *G* and *J* matrixes are built from the elements of the ones presented in (2.29) which correspond to the chosen state variables for this system as we show in the following equations. The *B* matrix depends on the location of the lateral thrusters in {*b*}, assume here to be symmetrical.

$$M = \begin{bmatrix} m+a_{11} & 0 & mz_G \\ 0 & m+a_{33} & -mx_G \\ mz_G & -mx_G & I_{yy}+a_{55} \end{bmatrix}$$

$$D = diag\{ D_{v_x}, D_{v_y}, D_{w_y} \}$$

$$G = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & z_G mg \end{bmatrix} \tag{2.34}$$

$$J = \begin{bmatrix} cos(\psi_0) & -sin(\psi_0) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ z_{port} & x_{port} \end{bmatrix}$$

Having into account that the vehicle's centre of gravity has x_G coordinate different from zero, the real steady-state *pitch* angle will not be null as previously assumed when linearizing. The differences in the linear model come only in the *J* matrix that will have the following form:

$$J = \begin{bmatrix} cos(\psi_0) & -sin(\psi_0) & 0 \\ sin(\theta_0) & cos(\theta_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.35}$$

In spite of the new model obtained being very close to the real blimp's behaviour, this subtle difference is of enormous importance in terms of control design, especially in the case of the state variable *z*.

If we analyse thoroughly the influence of this nonzero *pitch* angle in the state variable *z,* we can see that this variable is the integral of two velocities, instead of the simple integral of the v_z velocity:

$$\dot{z} = sin(\theta_0)v_x + cos(\theta_0)v_z \tag{2.36}$$

This implied that the vehicle's vertical, expressed in {*w*}, is dependent on its *pitch* angle as well as the v_x velocity. This can lead to a case where, if v_x is sufficiently large, the v_z velocity even has the opposite direction of $\dot{z}$. This can have strange effects in terms of control as the control law sees this velocity as being the approximate integral of v_z, with a small component from v_x.

Furthermore, the influence of v_x in the value of *z* is of great importance, as is shown in the control design section, because, in spite of wanting to drive the v_z velocity to zero, the v_x

velocity is aimed at some constant value different from zero, in general. The altitude value z only stabilizes when the inputs of the integrator reach zero, which has to be the result of the annulment of the two terms shown in Equation (2.36) and, thus, the velocity v_z is not null.

3. Sensor Description and Modelling

Most control applications require a precise sensor, in order for the system to know its state and act accordingly. For this work, vision was chosen to play that role and, therefore, the sensor used was a very small wireless camera.

In this chapter, we begin by presenting some of the basic concepts of projective geometry and employ these to introduce the pinhole camera model, which is used throughout this work. The matrix notation used to describe the transformations is taken from [9] and [10].

This model is a simplification of reality and it is, therefore, not enough in cases where there is much radial or tangential distortion due to the existence of a lens, for example. Even so, the theory developed around this model is consistent and, as this deviation from the model can be corrected if the distortion parameters are determined by calibration [11], the model is, only then, used. The distortion is accounted for by using an extended camera model that requires camera calibration for estimation of the distortion parameters.

If the model's parameters are well known, it is possible to reconstruct camera pose and velocities from the homographies between camera and ground plane or consecutive camera planes. Such reconstruction is carried out in this work, and used for control.

3.1. The Wireless Micro Camera

The camera used is the MVC30A, a black and white CCD micro camera with microphone included, but which is not used. The whole camera electronics are about 30 mm square in size, as can be seen in Figure 3.1, and its focal length is 4.8 mm. The CCD matrix has a sensibility of 0.1 lux.

A radio transmitter was connected to the camera output in order to allow wireless transmition of the captured video stream.

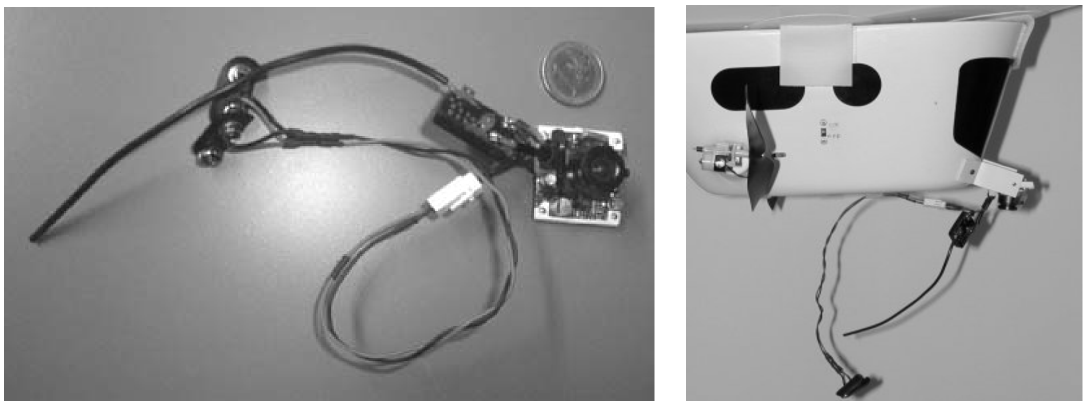


Figure 3.1 - The micro camera with transmitter attached and the camera support rig

3.2. The Pinhole Camera Model

The camera image plane contains a 2D projection of the surrounding 3D world. Any point $[X\ Y\ Z]^T$ in space is projected onto a point in the image plane $[x\ y]^T$. This point can be obtained by intersecting the image plane with the line that unites the 3D point and the camera optical centre *O*, corresponding to the origin of the camera frame {*c*}. This model is represented in Figure 3.2 and explained in further detail below.

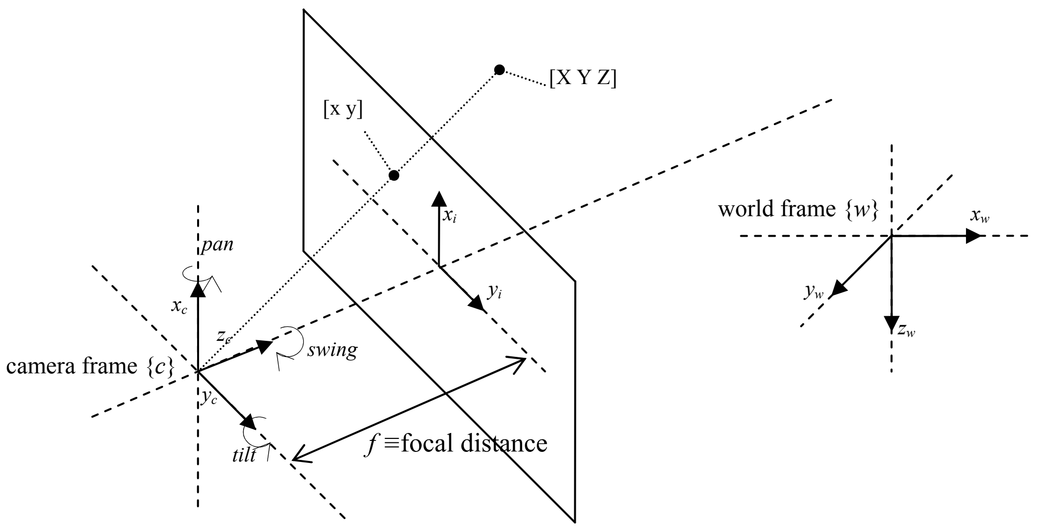


Figure 3.2 - The perspective camera model

If we express the world and image points' coordinates as homogenous vectors, the mapping performed by the camera can be expressed as a linear mapping from 3D space to the 2D image plane. Introducing $P_{camera}=[X\ Y\ Z\ 1]^T$ as the representation of any world point described in the camera reference frame and $P_{image}=[\lambda x\ \lambda y\ \lambda]$ as the corresponding point in the image plane, there is a linear relation between these vectors, which we show here in (3.1).

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} fx & 0 & 0 & 0 \\ 0 & fy & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

If one desires to place the origin of the image frame in the lower left corner of the image instead of the centre the previous equation comes affected by a matrix transformation. This transformation represents the displacement of *cx* pixels in the x_c direction and *cy* pixels in the y_c direction (where *cx* and *cy* represent the coordinates of the image centre in the new frame). Furthermore, the 3D point may not be represented in {*c*} and it becomes necessary to transform it to the camera frame in order to apply Equation (3.1). If we introduce $P_{world}=[X_w\ Y_w\ Z_w\ 1]^T$ as the representation of any world point in the world frame {*w*} and ${}^c_wR$ and ${}^ct_w$ as the representation of the attitude and position of the camera in the world frame, the final equation that describes the projection is:

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = K\left[I\,|\,0\right]\left[{}^c_wR\,|\,{}^ct_w\right] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \quad K = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

There is an alternative camera coordinate system representation, which is very useful in that it allows ignoring the intrinsic parameters of the camera. The normalized camera coordinates are obtained by imposing $X_w/x=Y_w/w=Z_w$ and this results, in practise, in the adoption of an equivalent camera with fixed unitary focal distance [12]. Point coordinates in the normalized camera image $[x_{norm}\ y_{norm}]$ relate to the normal camera image points $[x\ y]$ by the following equation (with K being the camera's intrinsic parameters matrix). The intrinsic parameters calibration is addressed in Appendix A.

$$\begin{bmatrix} \lambda\, x_{norm} \\ \lambda\, y_{norm} \\ \lambda \end{bmatrix} = K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (3.3)$$

3.3. Camera Kinematics

The camera is placed onboard the blimp and is the only sensor to be used for the control loop. We chose the location of the camera to be in front of the gondola, facing forward and down, with variable *tilt* angle, chosen a priori by the operator of the vehicle, null *pan* and *swing* angle of *π/2*. This enables the study of the performance of the algorithms with different *tilt* angles. The reference frame of the camera is further detailed in Figure 3.3.

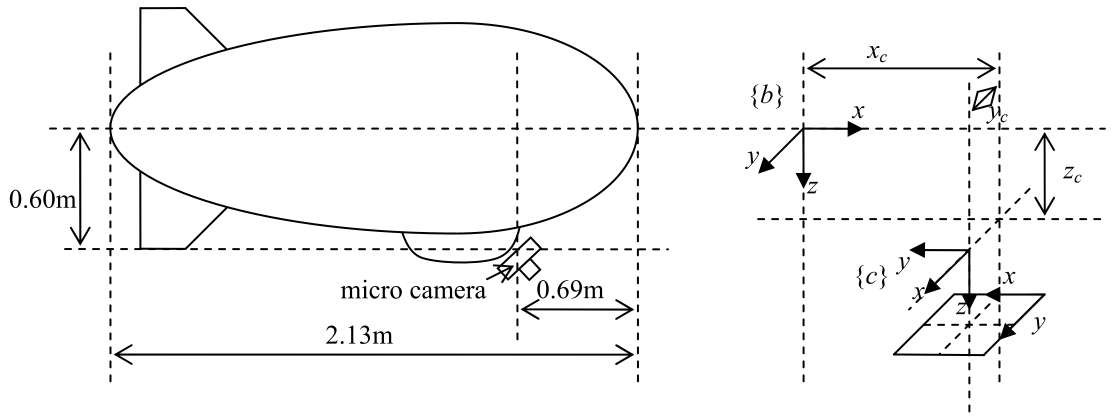


Figure 3.3 - Location of the camera frame with null *tilt*

The camera frame {*c*} pose is, therefore, related to the blimp's frame by the following transformation (using *pan=α*, *tilt=β* and *swing=γ* as defined in Figure 3.2).

$${}^{b}_{c}T = \begin{bmatrix} {}^{b}_{c}R & {}^{b}_{c}t \\ [0]_{1\times 3} & 1 \end{bmatrix}, \quad \begin{cases} {}^{b}_{c}R = \begin{bmatrix} -c_\gamma c_\beta & -s_\gamma c_\beta + c_\gamma s_\beta s_\alpha & s_\gamma s_\theta + c_\gamma s_\beta c_\alpha \\ s_\gamma c_\beta & c_\gamma c_\beta + s_\gamma s_\beta s_\alpha & -c_\gamma s_\theta + s_\beta s_\gamma c_\alpha \\ -s_\beta & c_\gamma s_\alpha & c_\beta c_\alpha \end{bmatrix} \\ {}^{b}_{c}t = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T \end{cases} \qquad (3.4)$$

This matrix represents the camera's extrinsic parameters, which are essential in rebuilding the blimp's pose as it is shown in Section 3.6. A simple procedure for finding these parameters is presented in Appendix A as well as some calibration results.

Note that the camera reference frame's *x* and *y* axes are parallel to the image *x* and *y* axes, which simplifies the structure of the intrinsic parameters' matrix *K*. In order for this to be, the camera has an offset in the *swing* angle of *π/2* for {*c*} so that it is naturally in the position shown in Figure 3.3.

We refer to the parameters ${}^{b}_{c}R$ and ${}^{b}_{c}t$ as the camera's extrinsic parameters which can be interpreted by substituting {*w*} by {*b*} in Equation (3.2). These parameters are time invariant and their calibration can be performed a priori, as explained in Appendix A.

3.4. Projective Geometry and Homographies

In this work, one major assumption was made about the structure of the robot's operation environment: we assume the floor to be locally planar and, therefore, it can be fully described by a 2D image map. The use of robust statistics in the vision algorithms allows for copping with slight deviation of the planar assumption [3].

Under this assumption, the projected image in the camera plane is related to the floor-map image by a one to one point relation. If we make the floor-map image coplanar with the *xy* plane, the Z_w coordinate of any world point is null. Redefining the coordinates of the points in the world as a 2D homogeneous vector, $P_{world}=[X_w \ Y_w \ 1]$, we can further simplify the equation presented above and arrive at the usual perspective projection representation.

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = K\,{}^{c}_{w}P \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix} \Leftrightarrow P_{image} = {}^{c}_{w}H\,P_{world}\ ,\quad {}^{c}_{w}P = \begin{bmatrix} \begin{bmatrix} {}^{c}_{w}r_{11} & {}^{c}_{w}r_{12} \\ {}^{c}_{w}r_{21} & {}^{c}_{w}r_{22} \\ {}^{c}_{w}r_{31} & {}^{c}_{w}r_{32} \end{bmatrix} & {}^{c}t_{w} \end{bmatrix} \tag{3.5}$$

The homography ${}^{c}_{w}H$ maps points from the world plane to points in the image plane, always expressed in homogenous coordinates. This 3×3 matrix is defined up to a scale factor and thus has eight degrees of freedom [12].

The considerations made before do not apply solely to the described setup but have a much broader use. Equation (3.5) can, thus, be rewritten in a more general way, representing the transformation from points in any 2D planar surface to points in any other 2D planar surface.

$$x' = Hx \tag{3.6}$$

It is, therefore, possible to represent the transformation from one image plane to the next (or any other, for that matter) by a homography. If we represent the image points in homogenous coordinates, we have (using *c1* and *c2* for the two camera's frames):

$$P_{image2} = {}^{c2}_{c1}H\,P_{image1} = {}^{c2}_{w}H\,{}^{w}_{c1}H\,P_{image1} = {}^{c2}_{w}H\,{}^{c1}_{w}H^{-1}\,P_{image1} \tag{3.7}$$

3.5. Homography Estimation from Image Features

There are many algorithms purposed in literature for the matching of image pairs and subsequent inter-image homography estimation. These can be divided into three groups, feature base algorithms, optical flow algorithms and gradient or minimization algorithms. It was not the objective of this project to develop such algorithms and therefore we relied on some previously studied procedures, developed by Nuno Gracias [3]. These algorithms rely on feature matching, either in the sequential filmed images or in the current filmed image and known floor-map image.

The matching algorithm comprises 3 steps: feature detection in one image, local feature matching in the images and finally outlier rejection from the locally paired features.

Features are interest points, usually chosen for having significant gradient values in more than one direction, like corners, being relatively invariant to image movement. These features are extracted from one image using a simplified version of the corner detector proposed by Harris and Stephenson.

Feature matching is done by cross correlation between the surrounding areas of each feature in one image and a small window around each feature in the other image. Using the Sum of Squared Distances (referred to as SSD) as a metric, the best matches are found, with sub pixel accuracy, and the pairs are kept.

The outlier rejection process is done by estimating the points' movement model and then performing a modified Least Median Squares (LMedS) algorithm to the results of applying the model to the paired points, assuming that outliers are not structured. The movement models used can go from a simple similarity projection with 4 degrees of freedom accounting for rotation, translation and scaling, to the more general planar projection of 8 degrees of freedom. This, in turn influences the time required to produce the homography results, and accuracy of the results.

The inter image homography estimating routines perform faster than the image to floor-map counterpart. This allows them to be used in the loop for the control of a slow moving vehicle, as large image overlap is desirable. These homographies only produce displacement estimation and not absolute location of the image according to the known world data, the floor-map. These can be considered as providing simple odometry, which leads to increasing position estimate error because of integration and therefore we need another way to estimate pose.

The image to floor-map algorithm takes longer to find a match, as expectable because it searches for features and matches in larger areas, but provides us with a global positioning method. It is used with a slower sample time to reduce the odometry resulting error. Image size is a very important factor as there will be a compromise between resolution (and therefore, precision in the matching) and processing speed.

In the real system loop, the fastest process of inter-image homography estimation, performs at about 13Hz (about 78ms) in parallel with the other global matching routines, for images sized 192x144 pixels. The processing is done in a Dual Pentium machine at 800MHz.

3.6. Pose Reconstruction by Homography Decomposition

It is possible to obtain relative rotation and translation between camera frames or from the camera to the world frame, just by analysing the homographies that describe the transformation between the corresponding planes. This fact allows the vision sensor to attain the system's full state, as long as the camera's parameters (intrinsic and extrinsic) are known with some precision. There are two different problems in such decomposition depending on the kind of homography we are studying.

Firstly, let us analyse the case of the homography that maps the floor plane into the image plane. As shown in (3.5) this homography matrix is the result of the camera's intrinsic parameters matrix and its pose relative to the 3D world plane. We can, therefore, easily recover the pose if the camera's intrinsic parameters are accurately calibrated. Firstly the normalized camera needs to be obtained by applying K^{-1} to (3.5).

Now, observe that the two left columns of ${}^{c}_{w}P$ are the columns of the rotation matrix of the camera frame apart from a scale factor. By looking carefully at the structure of the rotation matrix, it is clear that the scale factor comes:

$$c = \sqrt{{}^{c}_{w}p_{31}^{\,2} + ({}^{c}_{w}p_{11}^{\,2} + {}^{c}_{w}p_{21}^{\,2})} \tag{3.8}$$

By scaling ${}^{c}_{w}P$ using *c*, it is easy to obtain the pose of the camera in {*w*}, ${}^{c}_{w}T$, thus making it possible to reconstruct the movement of the camera in the 3D world.

Note, however, that the decomposition problem presents two solutions, depending on the side of the floor plane that the camera is in. It is therefore necessary to test the solution for the altitude value outputted and, if it is not satisfactory (it should be negative, as we are above the ground plane), apply the following transformation to the resulting matrix:

$$ {}^{c}_{w}T_{new} = \left[\begin{array}{c|c} {}^{c}_{w}R \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{matrix} {}^{c}x_{w} \\ {}^{c}y_{w} \\ -{}^{c}z_{w} \end{matrix} \end{array} \right] \tag{3.9}$$

Using the last algorithm, velocity estimates can be easily obtained, as we have the poses for the vehicle. But, when the image to floor-map match is done, the position estimate may suffer a large jump as the position error is zeroed, which causes jumps in velocity estimates. This is why it is useful to decompose the inter image homography in order to reconstruct displacement and not velocity from consecutive poses. This decomposition problem is called scaled Euclidean reconstruction. The decomposition algorithm used is presented in [12] and also relies on the fact that the normalized camera can be obtained with precision. Using Equations (3.5) and (3.7) it is possible to show that, being *n* the floor plane's normal vector and *d* the distance from the first camera to the plane:

$${}^{c2}_{c1}H = KAK^{-1}\,, \quad A = \left[d\,{}^{c1}_{c2}R + {}^{c1}t_{c2}n^{T} \right] \tag{3.10}$$

This homography structure can be decoupled and the displacement and rotation obtained from *A*, through Single Value Decomposition (SVD) and subsequent analysis of the solutions, which are, again, not univocal.

It is now important to establish the main limitation of these decomposition algorithms, related to the model used in the homography estimation. Depending on the model used, the homography contains different information about the movement of the camera and, for example, with full planar homography estimation, the reconstructed pose is ideally the real one but, using only a similarity homography model, even with perfect calibration, the real pose cannot be obtained. This is due to the fact that the *roll* and *pitch* angles of the camera are considered to be zero because of the homography matrix's resulting structure. On the other hand, the similarity model is much faster to obtain and the resulting motion estimates are less noisy than the ones provided by the planar model.

4. Control Design for the Blimp

The vehicle used in this work has highly non-linear dynamics and presents interesting challenges in terms of control, especially due to the configuration of the actuators and the particular characteristics of the camera sensor.

The operation of the blimp is done, in general, at low speed and its attitude is predominantly the one contemplated in the linearized models. Thus, the well-known and much proved methods from linear control theory have a very important role in the analysis and design of control for this system. On the other hand, there are couplings between state variables and other non-linear effects that grow larger the further we are from the linearization conditions. Considering this, the advantages of non-linear control are larger, especially in the presence of unmodelled system dynamics or sensor errors.

Finally, depending on the way that the image processing associated with the camera is performed, the system we are controlling can also change. As explained before, the sensor we are using ideally allows us to access the whole state of the system. In spite of that, due to processing speed compromises, it is possible that the estimated state variables do not include the full state (if a homography description simpler than full planar is used) and therefore the presented decoupled systems can be further simplified, but loosing some state information.

4.1. Control Methodologies Used

In this section, a short run down of the theory behind the controllers designed is presented. For a more detailed study of the algorithms proposed report to [5, 14, 15, 16].

4.1.1. Linear Control (LQR)

The sensor used, vision, in the case of a well-calibrated camera allows us to obtain fairly reasonable values for the state variables in spite of the associated processing noise and errors. Having access to the system's state, it is desirable to employ linear controllers that make use of that knowledge. Thus, in this line of reasoning, a linear state variables feedback controller is implemented, making full use of the state variables information [5, 13, 14].

If we include the reference directly in the state variables and define the control as the result of a gain matrix applied to the measured state variables we have:

$$u = -K\tilde{x} \;, \quad \tilde{x} = x - x_d \tag{4.1}$$

$$\dot{x} = Ax + Bu \Leftrightarrow \dot{x} = (A - BK)\tilde{x} \tag{4.2}$$

In this way, the dynamic behaviour of the system whose dynamics are described in matrix A is conditioned by the values of the gain matrix K use in the feedback loop, as the system's poles will be defined by the following equation:

$$det(sI - A + BK) = 0 \tag{4.3}$$

The gain matrix, *K*, can be defined using optimal control by state feedback and its values determined by the solution of the Linear Quadratic Regulator (LQR) problem. The LQR methodology allows us to, in an intuitive way, assign weights to state variables' errors and actuation, according to our own interpretation of their importance in the performance of the system. The resulting cost function to minimize is defined as:

$$J = \frac{1}{2}\int_{o}^{\infty}(\tilde{x}^T Q\tilde{x} + Ru)dt\ , \text{ with} \begin{cases} Q = Q^T \geq 0 \ \ e \quad R = R^T > 0 \\ \tilde{x} = x - x_d \end{cases} \tag{4.4}$$

The matrix *K* is determined by finding the solution of the Algebraic Riccati Equation:

$$PA + A^T P - PBR^{-1}BP + Q = 0 \rightarrow K = R^{-1}B^T P \tag{4.5}$$

The state variables are chosen according to the system and the errors are directly introduced in the feedback of the state variables. The state can be extended to include an integrator if the necessity arises.

Note, however, that if the homography in which the state estimate is based is not the most generic, this type of control may lead to worse results, since it does not convey all the information required for a full state measurement.

4.1.2. Non-linear Control (Robust Sliding Mode Control)

The system we are controlling has several non-linearities and their effect is stronger the further away we are from the steady state values used in linearization. Furthermore, the effects of non-linearities in the actuators were neglected. This makes the implementation of a non-linear controller essential to test the performance of a controller that can cope with the unmodelled dynamics in a safer and more robust way. The existing theory for the control of non-linear systems is not as vast and generic as theory developed for linear systems, but there are many useful and powerful results for our case [15].

The choice was made to use robust control and implement a sliding mode controller. This type of controller guarantees stabilization of the system and reference following in the presence of unmodelled dynamics of unknown structure and even other effects that don't depend directly on the state variables, as long as these differences from the model are limited.

We opted not to employ adaptive control as the sensor used presents systematic and cumulative errors so the parameter tuning would be severely affected. Furthermore, there are model uncertainties whose structure is not known to the full, such as the case of the coupling Coriolis matrix terms or actuator related non-linearities.

A generic non-linear system's model, with only one input, can be presented in the following fashion:

$$x^{(n)} = f(x) + b(x)u \tag{4.6}$$

Let us define a variable surface *s*, in the state space, that results from the state error.

$$s = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x}\ \ , \quad \begin{cases} \tilde{x} = x - x_d \\ \lambda > 0 \end{cases} \tag{4.7}$$

Note that, for $\dot{s} = 0$, the system behaves in a manner to eliminate the state variables error with speed defined by the constant *λ*. The goal of the controller is now to allow the system to attain the surface *s* (sliding surface) and, once there, to behave with the dynamics resulting from Equation (4.7) thus converging to the desired state (sliding mode).

We can express the objective of keeping the system in the sliding surface by defining a sliding condition, outside of *s*:

$$\frac{d}{dt}s^2 \leq -\eta\,|\,s\,|, \quad \eta > 0 \tag{4.8}$$

By accomplishing this, it means that the squared value of the distance to the surface is lower along all the systems trajectories, outside of *s*. By satisfying this sliding condition, it is assured that all the system's trajectories are directed towards *s* and, once on it, the error dynamics behaves according to what is specified in the definition of the surface.

It can be proven that the convergence of the system towards the surface is done in finite time depending, essentially, on the value of *η* [15].

Take, as an example, the case of a generic second order non-linear system.

$$\ddot{x} = f(x) + b(x)u \tag{4.9}$$

It is assumed that the non-linearities present in *f(x)* and *b(x)* are not completely known but the modelling error is limited.

The sliding surface is defined and its derivative is presented:

$$s = \dot{\tilde{x}} + \lambda\tilde{x} \Rightarrow \dot{s} = \ddot{\tilde{x}} + \lambda\dot{\tilde{x}} = \ddot{x} - \ddot{x}_d + \lambda\dot{\tilde{x}} = f + bu - \ddot{x}_d + \lambda\dot{\tilde{x}} \tag{4.10}$$

Therefore, so that we can guarantee that $\dot{s} = 0$ the form of the control must be the following (in which the terms related to the known model of the system are included):

$$u = \hat{u} + u' = \frac{1}{\hat{b}}(-\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}}) + u' \tag{4.11}$$

$$\Rightarrow \dot{s} = f - \frac{b}{\hat{b}}\hat{f} + \frac{b}{\hat{b}}\ddot{x}_d - \frac{b}{\hat{b}}\lambda\dot{\tilde{x}} - \ddot{x}_d + \lambda\dot{\tilde{x}} + bu' = (f - \frac{b}{\hat{b}}\hat{f}) + (1 - \frac{b}{\hat{b}})(-\ddot{x}_d + \lambda\dot{\tilde{x}}) + bu' \tag{4.12}$$

Now we can define the control portion that was still unknown and apply it to (4.12):

$$u' = -\frac{1}{\hat{b}}k\,sgn(s) \Rightarrow \dot{s} = (f - \frac{b}{\hat{b}}\hat{f}) + (1 - \frac{b}{\hat{b}})(-\ddot{x}_d + \lambda\dot{\tilde{x}}) - \frac{b}{\hat{b}}k\,sgn(s) \tag{4.13}$$

Using the condition found for $\dot{s}$ in (4.8) the gain *k* is limited in its lower value by a condition that allows the system to fulfil the sliding condition.

$$\frac{d}{dt}s^2 = \dot{s}s = (f - \frac{b}{\hat{b}}\hat{f})s + (1 - \frac{b}{\hat{b}})(-\ddot{x}_d + \lambda\dot{\tilde{x}})s - \frac{b}{\hat{b}}k\,|\,s\,| \leq -\eta\,|\,s\,| \tag{4.14}$$

$$k \geq \left| \frac{\hat{b}}{b} f - \hat{f} + (\frac{\hat{b}}{b} - 1)(-\ddot{x}_d + \lambda \dot{\tilde{x}}) \right| + \eta \frac{\hat{b}}{b} \geq \frac{\hat{b}}{b} | f - \hat{f} | + \eta \frac{\hat{b}}{b} + \left| \frac{\hat{b}}{b} - 1 \right| \left| (\hat{f} - \ddot{x}_d + \lambda \dot{\tilde{x}}) \right| \qquad (4.15)$$

The tuning of the constant *k* allows us to assure the convergence to *s* and therefore the following of references by the controlled system. This gain represents, in a way, a compromise between system robustness to unmodelled non-linearities and performance.

If we add a constant gain ($-K_d$, with $K_d>0$) applied directly to *s* we can improve the speed of convergence of the system. Finally, the full control law from this method is:

$$u = \frac{1}{\hat{b}}(-\hat{f} + \ddot{x}_d - \lambda \dot{\tilde{x}}) - K_d s - \frac{1}{\hat{b}} k \, sgn(s) \qquad (4.16)$$

There is a setback, however, which results in a constant control activity when the system approaches the sliding surface, in the state space. The way to avoid this instantaneous change in the control signal caused by the switching part of the control law related to *sgn(s)* this term is replaced by a smoother function that softens this transition between negative and positive actuation within a radius σ of the sliding surface.

$$u' = -\frac{1}{\hat{b}} k \tanh(\frac{s}{\sigma}) \qquad (4.17)$$

The change in this term leads to a low-pass dynamic behaviour for the sliding mode, within this interval, thus eliminating the unwanted effects of the switching demanded by the previously defined control law, without altering the effects of the control significantly.

4.2. *Yaw* Control – Heading System

For control purposes the vehicle is assumed to be non-holonomic (in spite of the fact that there is undesired actuation in the *y* direction, due to the geometric setup of the stern thruster, and thus drift in that direction) and we intend to control the *yaw*. Because of this, the position error in *y* should not be taken into account in the design of any control system. If we were to control the *y* position as well as the *yaw* the control action related to the error in *y* would influence the convergence of the *yaw* directly (and generally in opposition). Furthermore, variations in the value of *y* are expressed in world coordinates and, therefore, depend directly on the *yaw* angle, so the system is non-linear and time-variant in this variable.

Yet the v_y velocity can be directed to zero in order to avoid that the blimp should stray too much from its initial position while turning. Thus, the simplified time-invariant system model that is used is the same as in 2.4.1 but without the state variable *y*.

The system that is being studied here has an input $u=F_{stern}$, five state variables $x(t)=[v_y(t)\ w_x(t)\ w_z(t)\ \phi(t)\ \psi(t)]^T$ and the matrixes that define it come directly from Section 2.4.1 by eliminating the lines and columns corresponding to *y(t)*. Assuming that the sensor can provide the values of the state variables with some precision we opt to firstly design a LQR controller as described in 4.1.1.

The state error is defined as:

$$\tilde{x} = \tilde{x} - \tilde{x}_d = \begin{bmatrix} v_y & w_x & w_z & \phi & \psi - \psi_d \end{bmatrix}^T \qquad (4.18)$$

Having defined the weight matrix *Q* and the value for the scalar *R*, we obtain, by solving the Riccati Equation:

$$u = -K\tilde{x} \ , \quad \tilde{x} = x - x_d \tag{4.19}$$

This method presents us with a fairly simple control architecture but may encounter problems related to the unregarded non-linearities and couplings with the XZ system. The design of a more sophisticated and non-linear control law is therefore desirable in order to try to surpass these problems in a more specialized way.

Traditional robust control methods are not directly applicable to this problem because of the fact that the state variables' vector is multi-dimensional and the actuation vector is one-dimensional. To go around this problem, a robust controller is proposed in [5] and implemented by us, using robust control techniques as well as state feedback. This method is presented in a summarized way in the following paragraphs.

The system, with state variables $x(t)=[v_y(t)\ w_x(t)\ w_z(t)\ \phi\,(t)\ \psi(t)]^T$ can be represented under the general form of a linear system with an unknown non-linear term.

$$\dot{x} = Ax + Bu + f(x) \tag{4.20}$$

The control law we will apply to the system is made up of two parts, one corresponding to linear state variables feedback and the other with a non-linear form adapted to the non-linear part of the system and implementing the robust control terms.

$$u = u_l + u_{nl} \ , \quad u_l = -Kx \tag{4.21}$$

The system will, therefore, behave according to the following dynamics:

$$\dot{x} = (A - BK)x + Bu_{nl} + f(x) \tag{4.22}$$

We now define the sliding surface to where we will make the system converge. By assuring convergence to the surface, we guarantee reference following as well as stability.

$$s = H^T\tilde{x} \ , \quad \begin{cases} \tilde{x} = x - x_d \\ H^T \in \Re^5 \end{cases} \tag{4.23}$$

$$\dot{s} = H^T(\dot{x} - \dot{x}_d) = H^T(A - BK^T)x + H^T Bu_{nl} + H^T f(x) - H^T\dot{x}_d \tag{4.24}$$

Now we choose the non-linear part of the control law to be (assuming $H^T B \neq 0$):

$$u_{nl} = (H^T B)^{-1}\left(H^T\dot{x}_d - H^T\hat{f}(x) - k\,sgn(s)\right) \tag{4.25}$$

$$\Rightarrow \dot{s} = H^T(A - BK^T)x - k\,\mathrm{sgn}(s) + H^T(f(x) - \hat{f}(x)) \tag{4.26}$$

If we make sure that $A\text{-}BK^T$ has a null eigenvalue, we can choose H^T so that this vector is the corresponding eigenvector to that eigenvalue, we can therefore nullify the term of (4.26) that depends on this linear part of the system. Furthermore, if we choose the eigenvector related to the *yaw* state variable we are making sure that there is an integrator in the loop and so the tracking is guaranteed. And by using the sliding condition from Equation (4.8), we can find the lower limit for *k*.

$$\dot{s} = -k\, sgn(s) + H^T(f(x) - \hat{f}(x)) \tag{4.27}$$

$$\frac{d}{dt}s^2 = -k\, sgn(s)s + H^T(f(x) - \hat{f}(x))s = -k\,|\,s\,| + H^T(f(x) - \hat{f}(x))s \le -\eta\,|\,s\,| \tag{4.28}$$

$$k \ge ||H||.||f - \hat{f}|| + \eta \tag{4.29}$$

This control law can be proven to surpass the performance of the linear law defined in (4.19) if the non-linearities are more predominant [5]. In any case, these two control laws ensure system stability and allow the system to follow given references.

4.3.Control in the Vertical Plane - XZ System

As was said previously in Section 2.4.2 this system's full set of state variables is $x(t)=[v_x(t)\ v_z(t)\ w_y(t)\ x(t)\ z(t)\ \theta(t)]^T$ and for actuation we have $u=[F_x\ F_z]^T$. Yet, by analysing the system with care, we can see that the *x* position is dependent on the value of *yaw* in a non-linear way, as is the case of *y* (see Equation (2.34)). As the *yaw* angle is not constant and, in fact, is supposed to take any possible value, we cannot define a steady-state value for this angle in our system and therefore any linearization for *x* is not valid. That leads to the exclusion of this state variable from the ones we wish to control. However, it makes full sense to control the v_x velocity and that is one of the objectives of the control system.

The other objective is the keeping of a constant altitude *z* as well as trying to stabilize the other system variables. Note that, as explained in 2.4.2, the blimp's centre of mass has x_G coordinate different from zero and, thus, the steady-state *pitch* angle is not zero. Therefore, there is no sense in trying to regulate this variable to zero.

4.3.1. Coupled Control in the Vertical Plane

The simplified system that is used in the design of the control law has two inputs $u=[F_x\ F_z]^T$ and four state variables $x(t)=[v_x(t)\ v_z(t)\ w_y(t)\ z(t)]^T$. The matrixes that define it come directly from Section 2.4.2 by eliminating the lines and columns that relate to the unused variables *x* and *θ*. Robust control theory is not used in the full system as there is no direct method that could be applied to.

Assuming that the state variables are known with precision we opt to design a LQR controller, as described in 4.1.1, in order to make full use of the information from the sensor.

The error is defined as being:

$$\tilde{x} = \tilde{x} - \tilde{x}_d = \begin{bmatrix} v_x - v_{xd} & v_z & w_y & z - z_d \end{bmatrix}^T \tag{4.30}$$

In this case, the LQR controller design process even allows us to establish some sort of priority for the two actuation forces, making it possible to give more importance to the following of v_x references or maintaining the desired altitude *z*.

The fact that we control velocity, in the case of v_x, leads to the need of including an integrator in the loop. Furthermore, because of the influence of v_x in the altitude of the blimp, there is the need to compensate for this. In practise we are imposing that z acquire a constant reference value when at the same time we are trying to make v_x take some value, in general different from zero. As z is dependent on the integral of v_x, this leads to a variable steady-state gain in the altitude. What happens is that the controller output is not zero but the result of the combination of v_x and v_z in Equation (2.36) is a null value. Thus, it is necessary to place an integrator in the loop for this state variable, thus solving the gain problem. The system, redefined in order to contemplate these new state variables $x_I=[x_{Iv}\ x_{Iz}]^T \equiv [v_x/s\ z/s]^T$ is:

$$\begin{bmatrix} \dot{x} \\ \dot{x}_{Iv} \\ \dot{x}_{Iz} \end{bmatrix} = \begin{bmatrix} A & [0]_{4\times 2} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} & [0]_{2\times 2} \end{bmatrix} \begin{bmatrix} x \\ x_{Iv} \\ x_{Iz} \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u \ , \quad u = -\begin{bmatrix} K & K_{Iv} & K_{Iz} \end{bmatrix} \begin{bmatrix} \tilde{x} \\ x_{Iv} \\ x_{Iz} \end{bmatrix} \qquad (4.31)$$

As it is normal in any real control system, the actuators do not admit all actuation values, being subject to saturation either due to damping phenomenons or to voltage and current saturation of the motors' drive. This makes it necessary to provide the integrator placed in the loop with an anti-windup mechanism in order not to have the characteristic actuation delays brought on by this phenomenon. However, in this case, the solution does not come by just limiting the output of the integrator over the saturation value because the saturation will occur in the real thrusters, whose actuation value is one-dimensional and dependent on both F_x and F_z as is shown in Equation (2.32). Therefore, the anti-windup mechanism will be required to limit the integrator output only when the desired force to be applied by the thrusters F, reaches its saturation value.

The real coupling of these two virtual actuators leads to the necessity of taking special care in defining the reference values for velocity. The blimp has a maximum velocity value and any reference value larger than that will be impossible to attain. In spite of that, the control law will try to reach that v_x reference value and invariably impair the error convergence for the altitude reference in z.

4.3.2. Decoupled Control in the Vertical Plane – X System

It is possible to further simplify the system model used in the controller design, although in that manner we are, undoubtedly, eliminating some important dynamics. Yet, in this way, it is possible to come up with a representation of the system that addresses most of the damping related non-linearities that allows us to use the existing non-linear control theory to this problem. Therefore, we present a simplified non-linear model that describes the behaviour of the system in the state variable $x=v_x$ and has only one input $u=F_x$.

$$(m + a_{11})\dot{x} = -D_{v_x} x - D_{v_x v_x} |x| x + f(x,u) + u \qquad (4.32)$$

Due to the non-linear character of this model and all the unaddressed dynamic couplings with the other state variables, it becomes important to apply robust control methods, more precisely, sliding mode control for v_x, in this case.

We define the sliding surface having into account that the integral effect must be included in order to eliminate steady-state error and, therefore, there is a term that is related to the integral of the variable to control.

$$s = (\frac{d}{dt} + \lambda)\int_0^t \tilde{x}dr = \tilde{x} + \lambda\int_0^t \tilde{x}dr \qquad (4.33)$$

$$\dot{s} = \dot{x} - \dot{x}_d + \lambda\tilde{x} = \frac{1}{m + a_{11}}(-D_{v_x}x - D_{v_xv_x}|x|x + f + u) - \dot{x}_d + \lambda\tilde{x} \qquad (4.34)$$

Thus, in order to attain $\dot{s} = 0$ the control law must be (with big enough *k*):

$$u = D_{v_x}x + D_{v_xv_x}|x|x + (m + a_{11})(\dot{x}_d - \lambda\tilde{x}) - K_d s - k\,tanh(\frac{s}{\sigma}) \qquad (4.35)$$

As it is not possible do determine exactly the value for *k*, the tuning of this gain is left for the experimental part as well as the choice of the constant *σ* and gain K_d. Once again, it is necessary to eliminate windup phenomenons in the integrator and the architecture used here is the same as for LQR controller explained in the previous section.

4.3.3. Decoupled Control in the Vertical Plane – Z System

The reasons for designing a decoupled altitude control algorithm are similar to the ones presented previously for v_x control. The non-linear model used to represent the system is the following (with $x=z$ and $u=F_z$):

$$(m + a_{33})\ddot{x} = -D_{v_z}\dot{x} - D_{v_zv_z}|\dot{x}|\dot{x} + f(x,u) + u \qquad (4.36)$$

As for the case of the LQR, the influence of the nonzero *pitch* angle makes it necessary to include an integrator in the loop and so the sliding surface is defined as:

$$s = (\frac{d}{dt} + \lambda)^2\int_0^t \tilde{x}dr = \dot{\tilde{x}} + \lambda\tilde{x} + \lambda^2\int_0^t \tilde{x}dr \qquad (4.37)$$

$$\dot{s} = \ddot{x} - \ddot{x}_d + \lambda\dot{\tilde{x}} + \lambda^2\tilde{x} = \frac{1}{m + a_{33}}(-D_{v_x}\dot{x} - D_{v_xv_x}|\dot{x}|\dot{x} + f + u) - \ddot{x}_d + \lambda\dot{\tilde{x}} + \lambda^2\tilde{x} \qquad (4.38)$$

Therefore, for the system to converge to $\dot{s} = 0$, the control law must be the one presented below (with the tuning of the parameters left for the experimental part):

$$u = D_{v_z}\dot{x} + D_{v_zv_z}|\dot{x}|\dot{x} + (m + a_{33})(\ddot{x}_d - \lambda\dot{\tilde{x}} - \lambda^2\tilde{x}) - K_d s - k\,tanh(\frac{s}{\sigma}) \qquad (4.39)$$

4.4. Discretization of the Controllers

For simulation purposes, the continuous control laws are possible to test but, as the controllers are being implemented in a digital processor, the necessity arises to study the effects of this discretization. Furthermore, the simulation environment is supposed to be the most accurate representation of reality possible so, even in the simulator, the controllers implemented should be in their digital form.

The method chosen to discretize the control laws used is the Backward Euler, as it preserves controller stability and presents us with simple digital implementations of integrators or derivative control dynamics. These equivalences are shown here as implemented:

$$s = \frac{1-z^{-1}}{T} \quad \text{and} \quad \frac{1}{s} = \frac{T}{1-z^{-1}} \tag{4.40}$$

The main bottleneck that ends up determining the maximum possible operation rate is in the image processing algorithm, which, mainly, involves feature matching and thus correlation. This type of processing takes its time and, as was explained in Chapter 3, running the feature matching algorithms for the floor-map to image and the inter image homographies in parallel with the control and other routines leads to a sample time of around 78 ms in the processor we are using, for the 192×144 pixel images. This is the minimum sample time we can have with our experimental setup.

The analysis of the response speed of the system is made only for the linear system models, as these are a good approximation. Furthermore, this enables us to make our point in terms of frequency response and pole location simplifying the whole analysis process. The modelled system, separated in its two decoupled systems presents us with two different behaviours. The Heading system shows the fastest pole to be at around 7 rad/s with large imaginary part (responsible for the *roll* oscillations) and the XZ system shows the fastest poles to be at around 8 rad/s.

The Nyquist criterion tells us that the sampling frequency must be at least double the fastest system frequency in order for information not to be lost when sampling. In controller discretization, there is a rule of thumb, however, that tells us our sampling frequency should be about eight to ten times faster than the fastest system frequency [16]. Using the fastest pole to be at 8 rad/s.

$$T_s \simeq \frac{1}{k}\frac{2\pi}{w}, \quad k = 8 \text{ to } 10 \tag{4.41}$$

$$T_s = \frac{1}{k}\frac{2\pi}{w} \Leftrightarrow k = \frac{2\pi}{0.1\times 8} \simeq 7.8 \tag{4.42}$$

As shown, we are within a well acceptable proximity of this value and therefore the discretization method used presents us with good results, the discrete controllers not straying visibly from the continuous controllers' response.

5. Navigation

Having discussed the implementation of heading, velocity and altitude control, it is now possible to control the system to follow any given feasible reference. The problem in hand can be described as the definition of an error function for heading (*yaw*) and velocity value (v_x) and resulting error to the actual robot velocity. The control objective regarding altitude is the maintenance of its initial value and, therefore, the trajectories are defined in 2D, although references for z could also be introduced for full 3D movement.

As we using vision as the feedback sensor, there are two ways of defining the error, resulting from having the desired velocity vector in two different spaces. One way is to define the error directly in the image itself, in pixels and radians, and the other consists of defining the error, based on the 3D pose of the blimp in Cartesian space, i.e. meters and radians.

Whichever space the error is defined in, it is important to devise a way to obtain the desired trajectory by defining some error function. Thus, two trajectory following behaviours were implemented, either based on simple point-to-point methods, but using smooth trajectories. We have also implemented a station-keeping algorithm that is activated upon arrival at the destination point.

5.1. Error Definition: Image Based Versus 3D Based

When we are dealing with navigation, the more intuitive manner of defining the position error is to obtain the vehicle's position in Cartesian space and simply calculate the error as being the difference between the current and the desired 3D position. The error is made null when the robot's frame is positioned in the end-point.

As mentioned in Chapter 3.5, it is possible to determine the relative rotation and translation between consecutive locations of the blimp's frame or even from the blimp to the world frame, by decomposing the estimated homographies. However, this is only possible if the internal and external camera parameters are exactly known a priori. This poses a problem due to errors associated with the experimental measuring of the parameters (see Appendix A). To avoid the error due to the use of external parameters involved in 3D reconstructed it is preferable to define the velocity vector error in some other manner.

To surpass pose reconstruction errors we include them inside the loop and, thus, the second method is to define the error in the image plane, being the control objective to place the destiny point in the image centre. This way, the position error zeroed when the end-point is the centre of the filmed image. However, note that, when in station keeping, the objective of the controller is not to take the blimp's vertical to the desired 3D point, but to get the camera central point to there. This leads to the phenomenon showed in Figure 5.1.

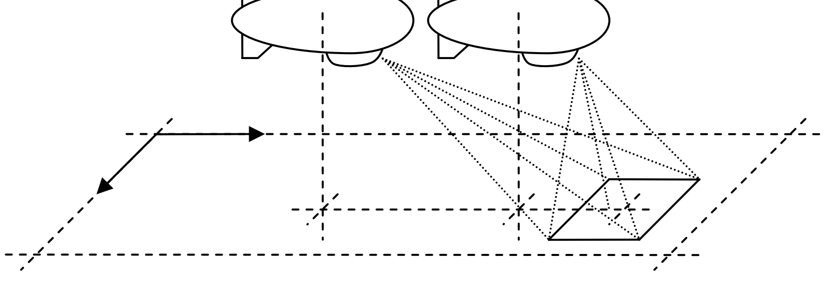


Figure 5.1 - Image based station keeping with different camera *tilt* angles

The larger the *tilt* angle of the camera and the vehicle's altitude, the further away from the final point the centre of the blimp will be. This is because the projection of the centre of the filmed image will be further apart from the projection of the blimp's frame vertical axis in the ground plane.

5.2. Point-to-Point Control and Trajectory Following

In both methods, the first step is the determination of the points that define the trajectory, in 2D world coordinates, i.e. in meters.

For point-to-point control, this ordered set of points is used as the destination. Each time that the vehicle arrives at a vicinity of the current way point, this is updated to the next and so the robot will proceed to the next point. This will happen at every way point that has been defined until the last one is reached, where the robot will perform station keeping.

In Cartesian space, a neighbourhood is defined for the points, in metric coordinates, and a line-of-sight sort of behaviour [5] is implemented, i.e. when the vehicle "sees" the current destination point it proceeds to the next,. In image space, the line-of-site behaviour is more intuitive as the point can be considered as being reached when it appears visible inside the filmed image. For the image plane references, the whole trajectory is transformed to the image plane before measuring these distances.

In the case of smooth trajectory following, it is necessary to connect the pre-defined set of ordered 2D points using a smooth curve describing a trajectory to be performed by the robot. For this, Splines curves were used. In each of the segments, a few points are chosen in order to cover the whole trajectory with equidistant points (distance configurable). A result for this is illustrated in Figure 5.2.

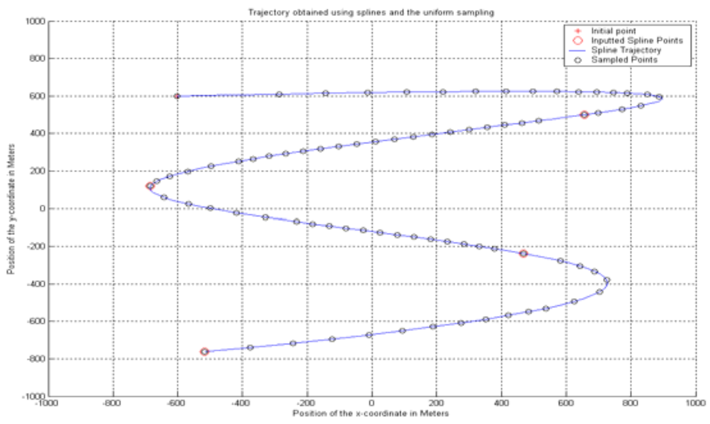


Figure 5.2 - Trajectory points from Spline curves

Using the new set of points (in which the original points are also included), we apply a modified point-to-point algorithm in which the destination point is made to be one that is a predefined number of points ahead (defined by the look-ahead parameter) from the current point . The current point is defined as being the closest point (using Euclidean distance) from a set composed by the previous point and some close next ones. When defining the destiny point in the image plane the distances are measured to the image centre by transforming the 2D world points to the image plane.

Having chosen the destiny point, either in 2D metric coordinates or in image pixels in the current camera image, its is necessary to describe the heading and displacement manoeuvres, i.e. the vehicle's velocity vector (norm and direction), that takes us towards the desired point, resulting in the *yaw* and v_x errors.

The error for the heading system is defined as the difference between the desired *yaw* that points the vehicle towards the destination and the current *yaw*. In 2D this is calculated by:

$$e_{yaw} = arctan\left(\frac{y_{desired} - y_{current}}{x_{desired} - x_{current}}\right) - yaw_{current}\ , \quad e_{yaw} \in [-\pi, \pi] \tag{4.43}$$

Where the result of the *arctan* function is defined between [-*π,π*]. Note that, working in the image plane, the current *yaw* is always *π/2* due to the placement of the camera frame and, therefore, there is no need to know the *yaw* in the world frame. The *x* and *y* current coordinates are made null since we are have converted the desired point to the image plane.

For the velocity value, the error is obtained in a way that is independent of the reference space used. Assuming that the maximum velocity of the vehicle is known, it is only a matter of defining a window function W(e_{yaw}) such that $0<W(e_{yaw})<1$ that, multiplied by a fraction of the limit velocity, results in the desired velocity.

$$v_{x_{desired}} = v_{x_{max}} W(e_{yaw}) \tag{4.44}$$

In this way we make certain that the robot will not deviate from trajectory as it will gain increasing linear velocity the further the destiny point is at its front. Furthermore, this ensures that the actuators will not saturate trying to achieve an impossible velocity value thus compromising altitude maintenance. The choice of window function used influences the behaviour of the blimp at every moment. Through experimentation, we found that the Blackman window performed the best.

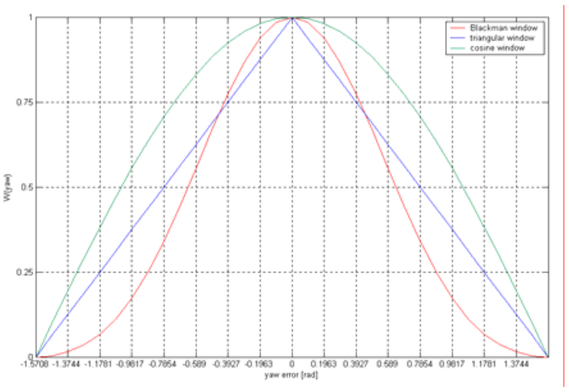


Figure 5.3 - Several windows possible for the calculation of the desired velocity

5.3. Station Keeping

The two methods of trajectory following discussed so far produce an output that takes the robot to perform a predefined trajectory. However, after this it is desirable that the robot, upon achieving the final point, maintains itself there, thus performing station keeping manoeuvres. Moreover, the approach of this end point should also be smooth in order to facilitate the station keeping procedure. When we are reaching the last trajectory point, the Euclidean distance between the destination point and the current position of the blimp (or the centre of the image plane, depending on the reference space) decreases. When this distance becomes smaller than a pre defined value it is assumed that the robot is in station keeping.

The way that the velocity error is defined is then changed, as the vehicle should slow down as it approaches the destination. Note this was not true in the first case were we did not use any information about the distance to the end-point. Again, the maximum linear speed value, multiplied by the window, is used. In addition, a non-linear function that depends on the distance to the point is used to force the v_x error to decrease. Also, note that if the point is towards the back of the blimp, it is logical that the approach is done backwards as the point should be near. Therefore, we check if the end-point is in front or behind the robot and the velocity sign is defined accordingly.

$$e_{vx} = \pm 1 \times v_{x_{max}} W(e_{yaw}) \tanh\left(\frac{\sqrt{(x_{current} - x_{end})^2 + (y_{current} - y_{end})^2}}{a} \right), \quad a \in \Re^+ \tag{4.45}$$

The *tanh* function is limited between 0 and 1 so it is, therefore, responsible for the gradual reducing of the desired velocity. The maximum velocity value is, again used. The constant *a* is an adjustable parameter that shapes the function and is used to adapt to image or Cartesian based references as well as for controlling the speed at which the vehicle slows down. Bellow we show the effects of changing this constant.

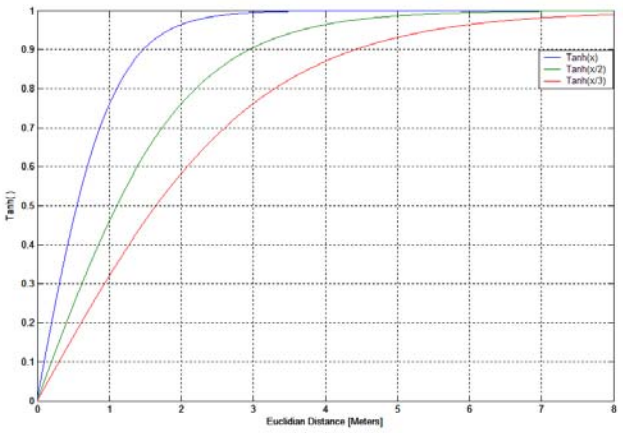


Figure 5.4 - Reshaping the non-linear function

Note that, depending on the value of *a*, there is the possibility of guaranteeing the absence of discontinuity in the desired velocity values when entering station keeping

In station keeping, the error definition for the heading system is also altered, in order to provide the robot with the capability of move forward and backward, instead of always moving forward. If the destination point is located forward, the heading error is given by (4.46) and if the point is located backward the heading error comes from (4.47).

$$e_{yaw} = \arctan\left(\frac{y_{desired} - y_{actual}}{x_{desired} - x_{actual}} \right) - yaw_{actual}, \quad \arctan(.) \in [-\pi, \pi] \tag{4.46}$$

$$e_{yaw} = \left(\arctan\left(\frac{y_{desired} - y_{actual}}{x_{desired} - x_{actual}} \right) + \pi \right) - yaw_{actual}, \quad \arctan(.) \in [-\pi, \pi] \tag{4.47}$$

This way, we choose the correct heading, in case the robot needs to go forward or backward. Finally, in order to ensure that the desired velocity vector (*yaw* error and v_x reference velocity) reaches zero, it is necessary to implement a small dead-zone in these variables. This allows for the system to reach a stable position and come to a stop, once there.

6. Experimental Results

Various tests were performed, in order to evaluate the quality of the algorithms developed and their robustness to poor camera calibration or model uncertainty. In this chapter, we first describe the experimental setups, the simulator and the real blimp with related hardware and software. Second, some of the most important tests are presented in order to assess the functioning of the image processing, controllers and navigation systems and show some of the inherent problems and qualities of the different variations possible.

6.1. The Experimental Setups

In spite of having the real experimental setup available, simulation is of vital importance in the building and testing of any control system. This is true, not only because it is easier to run the algorithms in the simulation than in the real system (mainly because of the preparation involved) but also because in the simulation environment we can have access to all the signals. Moreover, the real pose and velocity data, unavailable in the real system, which only outputs the camera image (sensor data affected by error), can be studied in order to evaluate the performance of the system.

The simulator mimics the real environment, but with added possibilities like the capability to bypass the sensor, test camera calibration or modelling errors, test the various estimation and control algorithms, alter thrusters' configurations or reposition the camera easily and rapidly. A more detailed explanation of the simulator is presented in Appendix B.

Having the whole system tested and working in the simulation environment it is essential to perform some testing of the algorithms in the real setup, as sample time limitations and clock jitter, transmission errors and other unexpected problems are very difficult to model and test in the simulator.

The physics of the blimp are already described in Chapter 2 and the camera is modelled in Chapter 3. It is now necessary to present a brief description of the support hardware for the communication between PC and the blimp and the software used for the image processing and control algorithms. A simple description of the whole system is shown in Figure 6.1.

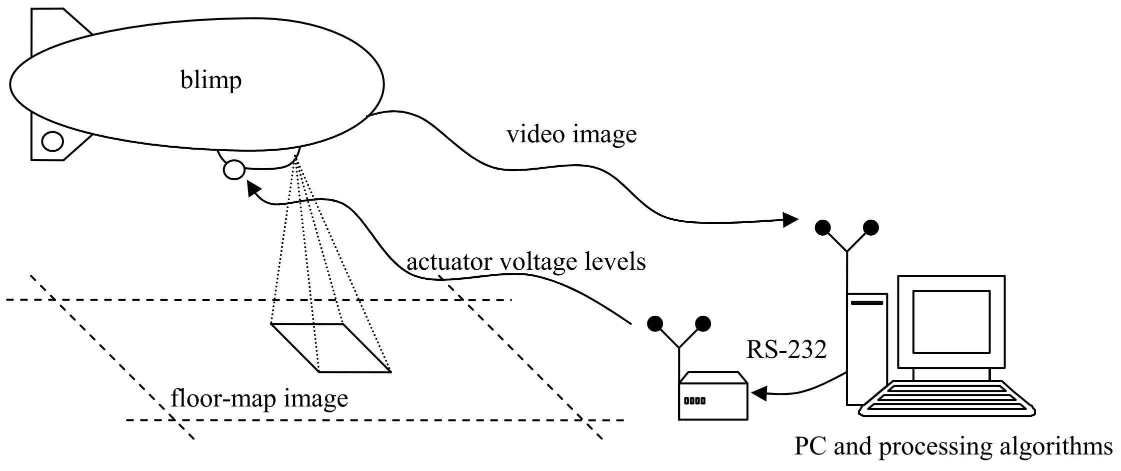


Figure 6.1 - The real experimental setup

The blimp was provided with a joystick-like wireless remote control. In order for the PC to be able to communicate with the blimp, the remote control was altered and an intermediate communication interface was assembled. Using the RS-232 protocol the PC sends the desired actuation levels from 0 to 255 to an external box that in turn is connected with the remote and therefore can output these values to the blimp's on-board receiver.

The micro camera has a radio transmitter attached, broadcasting images in the S37 channel. A Pinnacle® PC-TV receptor board with antenna is used to capture images in real-time. The PC we are using is a Dual Pentium® III 800MHz, with 512 MB RAM, running under Windows 2000 (service Pack 4).

The software used for image processing and control is based on the one used in [1, 2] in which image capturing and homography estimation was already implemented, in C++ language. This program, Viscon, is built in a modular way, basing all the different processing algorithms in independent threads. Therefore, in spite of having to do some modifications related to image capturing hardware and operating system changes, we essentially had to create the homography decomposing and controller thread, the actuation thread in which we inverted the thrusters' non-linearities and finally the RS-232 routines.

The image processing thread keeps the filmed images for future analysis and also a file where the data from the matchings and homography estimation is saved. The control thread also writes a file for each simulation where the reconstructed poses and velocities as well as actuation and reference values are presented so that the behaviour of the system can be easily analysed after each experiment. The same setup was used in the parameter identification experiments, presented in A.5.

6.2. Pose Reconstruction from Homographies

This experiment was done only in the simulation environment because it is the only way to compare estimation results with ground truth values, in a typical vehicle movement over the floor-map. The camera intrinsic and extrinsic parameters used are those of the real camera (see Appendix A).

We show a typical vehicle movement with small initial forward displacement, rotation starting at *t=4* s and, finally, saturation of the forward speed at *t=8* s. The results are presented below.

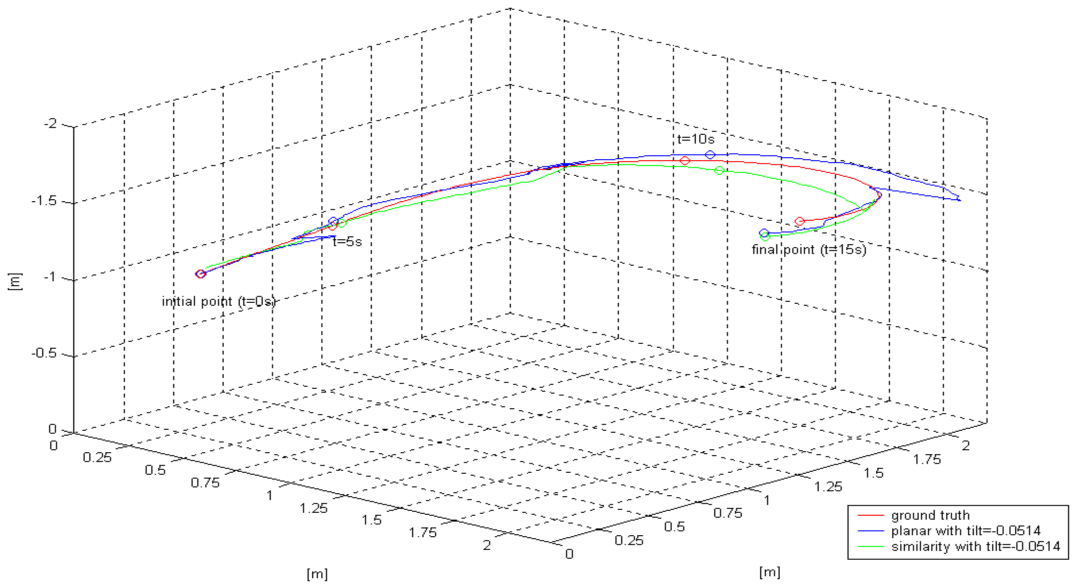


Figure 6.2 - Real versus estimated movement of the blimp's frame in 3D space

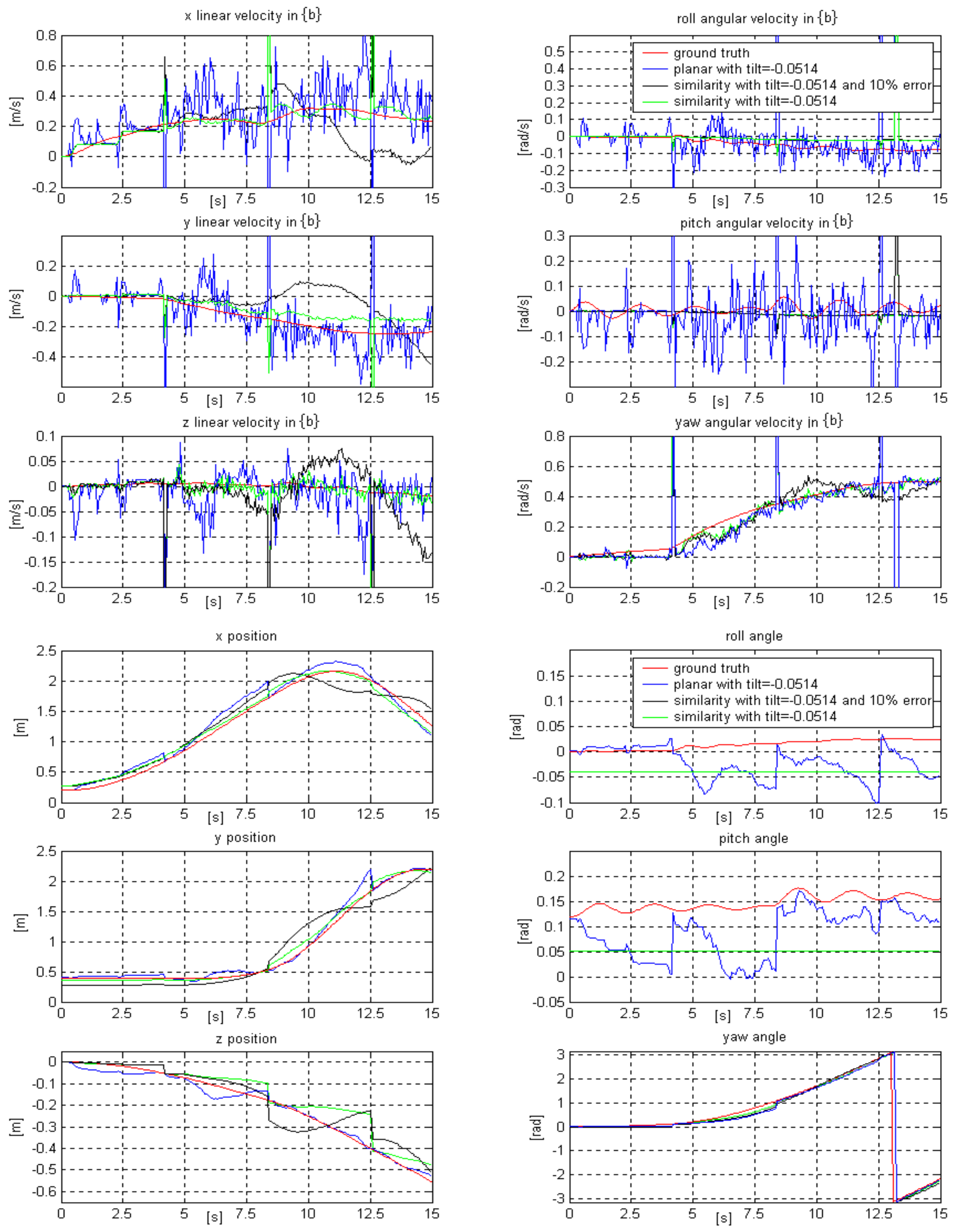


Figure 6.3 - Velocity and pose profiles for several algorithms and calibration errors

The use of the planar model results in very noisy velocity estimates and some error in *roll* and *pitch*, although the values are very small. These errors from the use of the planar method are explained by several factors, which can be the scarce texture in the floor image-map, added image noise and confusion between small angle variations and displacement. This last reason is the one that has the most effect, in this case, leading to the very irregular and inaccurate estimated velocity profiles shown and the errors in pose and attitude. A possible way to avoid this would be to perform some filtering on the signal, including knowledge about the robot's physics.

The similarity model leads to the best estimates, when the camera intrinsic parameters are known with precision, nevertheless, the *yaw* and *roll* are constant because these are not estimated and so the values come from the extrinsics' calibration directly. Velocity profiles from this algorithm are much smoother than the ones produced by the planar model, but can be influenced by large variation in the *pitch* or the *roll* angles, as the pixels in the image move forward or backwards accordingly and this is confused with linear velocity.

The difference from the perfectly calibrated camera and the one with error of around 10% in all intrinsic parameters results in linear velocity errors and bad altitude estimates. As shown in Appendix A, these errors never amount to this high level so these they should be of minor importance. Situations where errors in extrinsic parameters occur are not presented as these affect the output of the reconstruction directly, and so the results can be easily inferred.

6.3. Controller Step Response

In this Section, using the best performing controllers, a sequence of rotations and linear translations is performed, in order to test the controllers' ability to deal with the neglected couplings in the systems and the errors in the image processing system. Step responses performed with the real setup are also presented and analysed.

6.3.1. Simulation Results

As for the simulation results, firstly, the controller parameters were tuned through the performing of step responses using the various image processing algorithms. The influence of the image processing, as well as the resulting gains and weight matrixes determined by tuning are shown in Appendix C.

The best performing controllers (the LQR and the decoupled sliding modes for the XZ system and only the LQR for the Heading system) were employed in a more complex test including simultaneous rotation and translation and extrinsic camera parameters calibration errors. The dead-zone solution presented in Appendix C for removing the oscillations was used in these experiments, managing to reduce velocity oscillation amplitude in about 20%.

Figure 6.4 shows the movement with bypassed sensor, similarity estimation model and badly calibrated extrinsic parameters for the LQR controller.

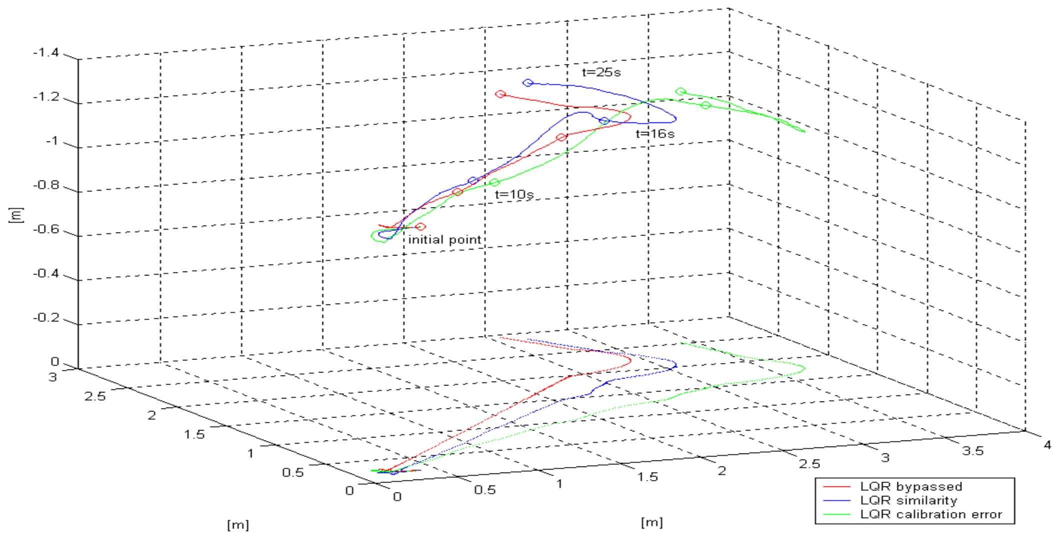


Figure 6.4 - 3D movement of the blimp's frame

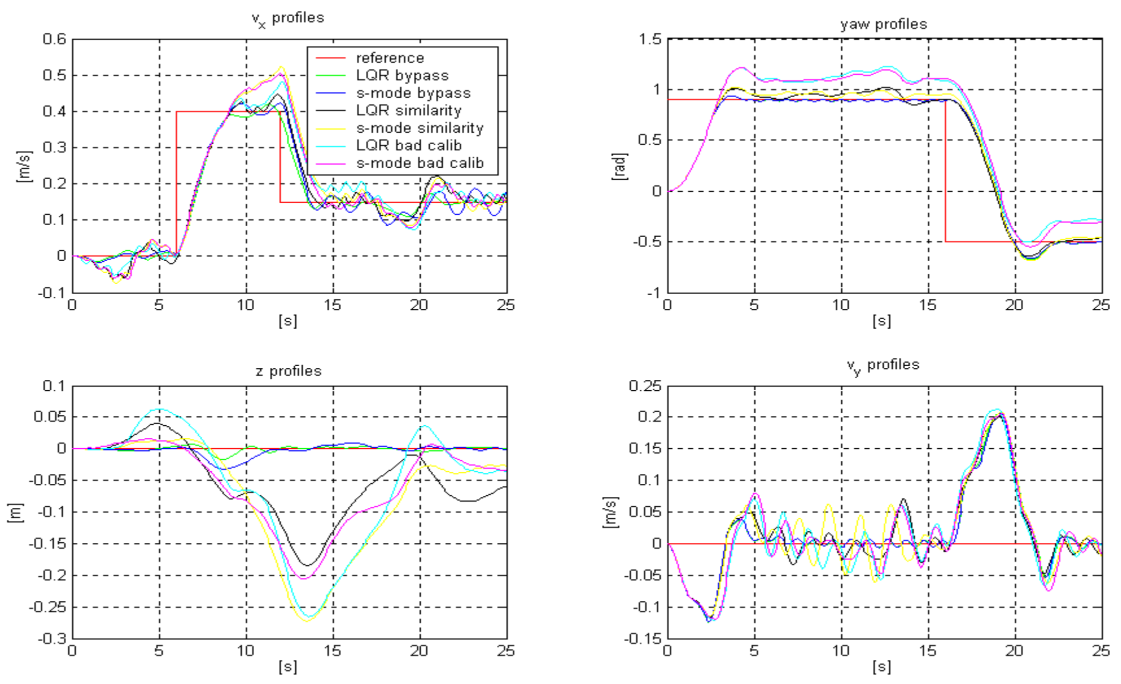


Figure 6.5 - Step response of the full system

The difference in the 3D movement happens, mainly, due to the *swing* angle error, which, in turn, leads to the error in *yaw*. As this error took longer to annul, the blimp drifted further in altitude. This was not perceived correctly by the homography algorithm, leading to an error in altitude, which affected velocity perception.

The altitude error does not reach values larger than 0.25 cm and this is considered satisfactory, taking into account all the errors introduced by calibration and the estimation algorithms. The altitude estimation error must be small, as it affects the calculations for the other state variables, especially the linear velocities v_x and v_y.

The error in v_x comes from the error in altitude estimation. If the distance to the floor is perceived as larger than it really is, the velocities are underestimated and the opposite occurs if the altitude is smaller than the one perceived.

In the case of the *yaw* the problem is more serious, as the error introduced in the extrinsic calibration of the camera leads directly to an error of around 0.2 rad (around 10% of $\pi/2$) in the yaw estimation.

The disturbance in v_x and z caused by the rotation of the blimp at t=16 is well handled by all the controllers and the desired values for the state variables are reacquired shortly after.

6.3.2. Real Setup Results

In the case of the real system, the experiments were performed only with the similarity model for the homography estimation, as it is the fastest and more reliable. The camera was mounted with small *tilt* angle and the LQR controllers were used.

Below, in Figure 6.6, a series of tests is shown, where the objective of the controllers was to maintain its initial altitude and null velocities and *yaw* angle.

The profiles presented are the result of pose and velocity reconstruction from the estimated homographies. This justifies the large oscillations in the perceived velocities, resulting, not only from the true oscillation in these, but also from large unobservable variations in the *roll* and *pitch* angles. In spite of a very oscillatory behaviour in v_x (larger than expected from the simulation results), the system manages to maintain null mean velocity.

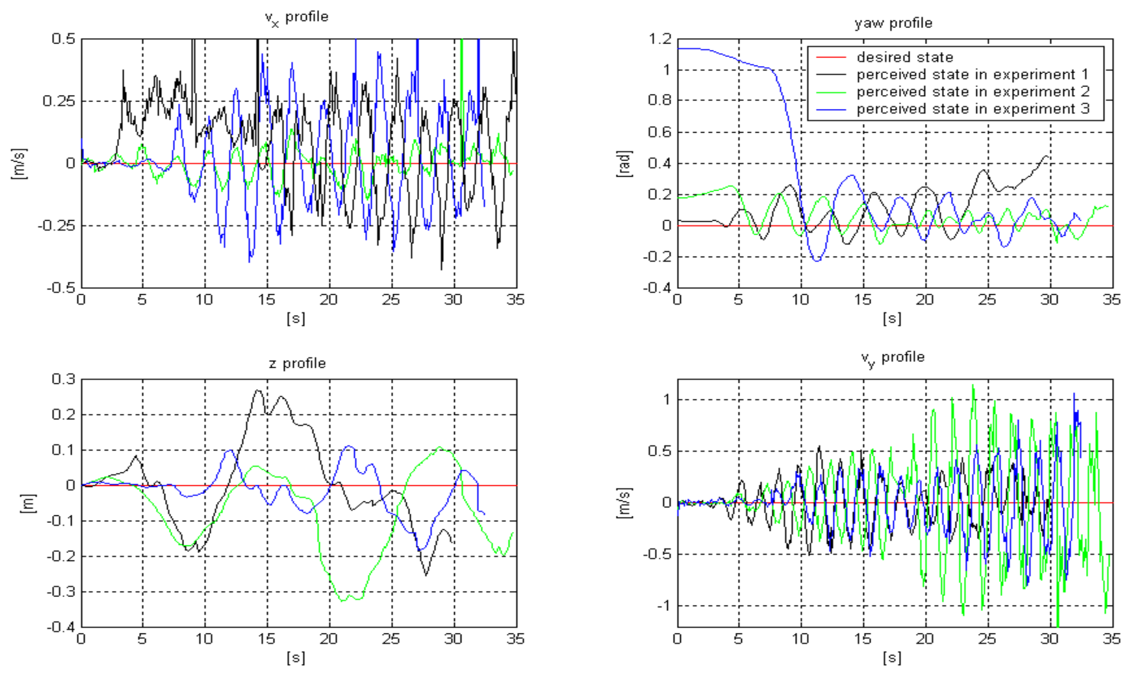


Figure 6.6 - Response of the real system to null reference

As for the *yaw*, the system manages to acquire the desired heading even from initial angles ranging to $\pi/2$, maintaining it around the desired value from there on.

Notice that, in black, an initial velocity was given to the vehicle, which, in spite of this, managed to stop and converge to the desired state.

After a small tuning in the controller gains, in order for the controller output not to be saturated all of the time, a slightly better behaviour was accomplished for the *yaw*. This is presented in Figure 6.7. The large oscillation in *yaw* at t=64 s was due to a disturbance introduced by us, where someone touched the tail of the vehicle inducing rotation, which was rapidly compensated and corrected by the controlled system.

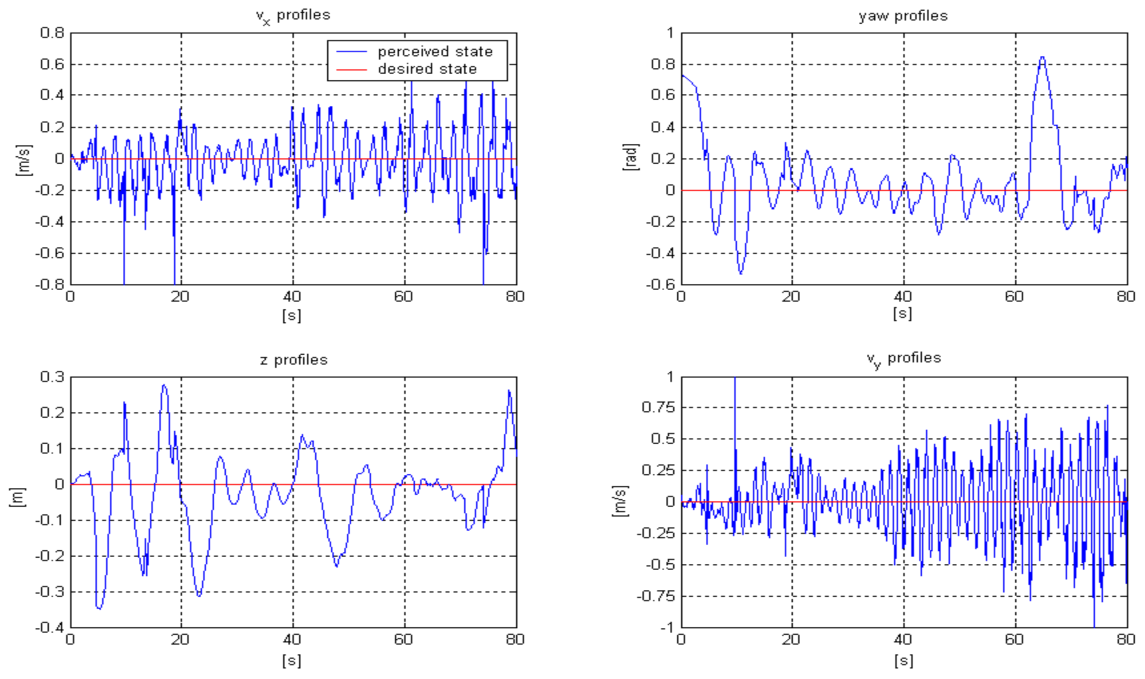


Figure 6.7 - Response of the real system to null reference (smaller gains)

In Figure 6.8 a second experiment is shown, where a velocity step is demanded from the system, which starts out with *yaw* different from the required one, thus having also to correct it, prior to advancing.

The system begins by reaching the desired *yaw* angle and then initiates the acceleration that leads it to the desired velocity. An initial peak in velocity around *t*=10 s is the result of this acceleration and subsequent variation in *pitch* angle. These variations in *pitch* angle are responsible for most of the oscillations perceived in v_x because of the use of the similarity model.

Between *t*=10 s and *t*=15 s the system acquired large error in *yaw* due to the effect of the asymmetries present in the tail fins which affect the behaviour of the vehicle when going forward. This is also why the velocity took longer to converge in velocity, than expected.

After reaching the desired velocity, around *t*=23 s, the blimp came to the edge of the image map, therefore having to be stopped manually. That is the reason for the sudden decrease and even inversion of the v_x velocity.

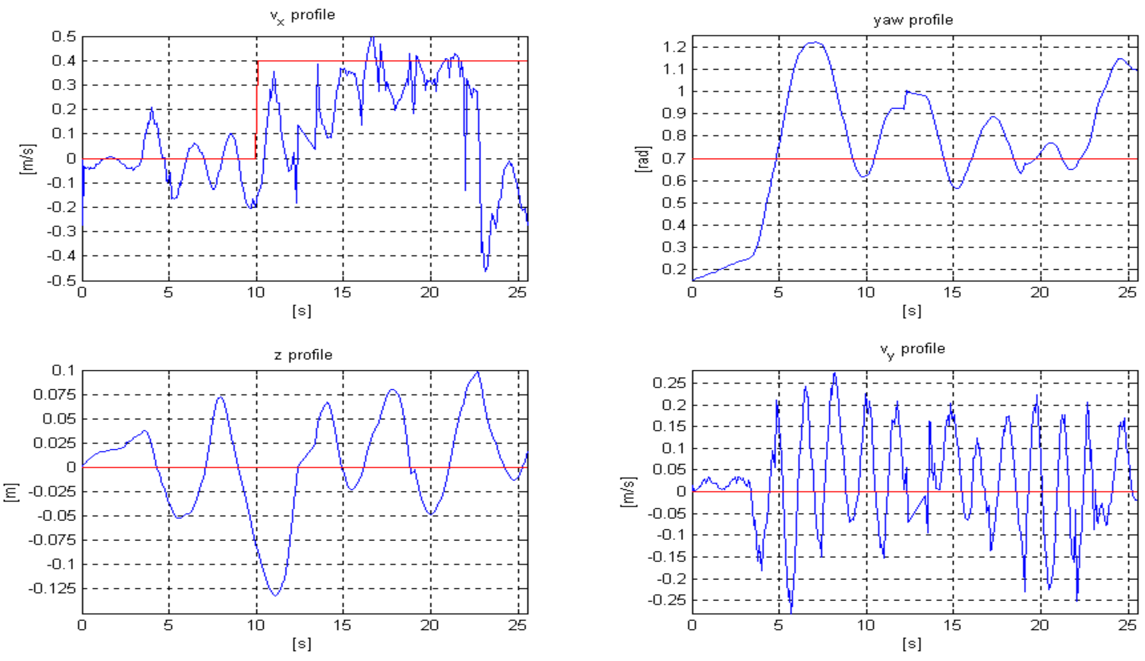


Figure 6.8 - Response of the real system to a forward velocity step

6.4. Reference Following

The reference following test presented in this section are the result of tests performed in the simulator. We start by bypassing camera so we can present a first comparison of the two possible reference definitions proposed. We then add the sensor and image processing routines, using the similarity model and, finally, these tests are repeated with errors in the extrinsic parameters' calibration.

The floor-map image over which the trajectory is made has a size of 4.8×3.6 m. The small size of the floor-map in relation to the blimp itself forced us to bound the maximal velocity to 0.4 m/s in all tests so that the blimp does not leave the floor-map during the experiments (the blimp's maximum velocity is arround 0.65 m/s).

The pre-defined trajectory used, was built so that it has 5 distinctive characteristics. It starts with a strait line followed by a smooth curve. At its end an s-type trajectory is desired, just before a tight curve that forces the blimp to turn a full 180 degrees. At the end of this curve a smooth line was implemented, crossing the initial trajectory line. The blimp took about6 55 s to perform these paths.

In all the figures, the blue line represents the pre-defined desired trajectory, the green one is the projection of the image centre in the floor plane and in red we show the trajectory performed by the origin of the blimp's frame projected on the floor plane..

In the first test, the system knows the exact pose and velocity of the blimp, available because the sensor is bypassed. Below we show the results obtained.

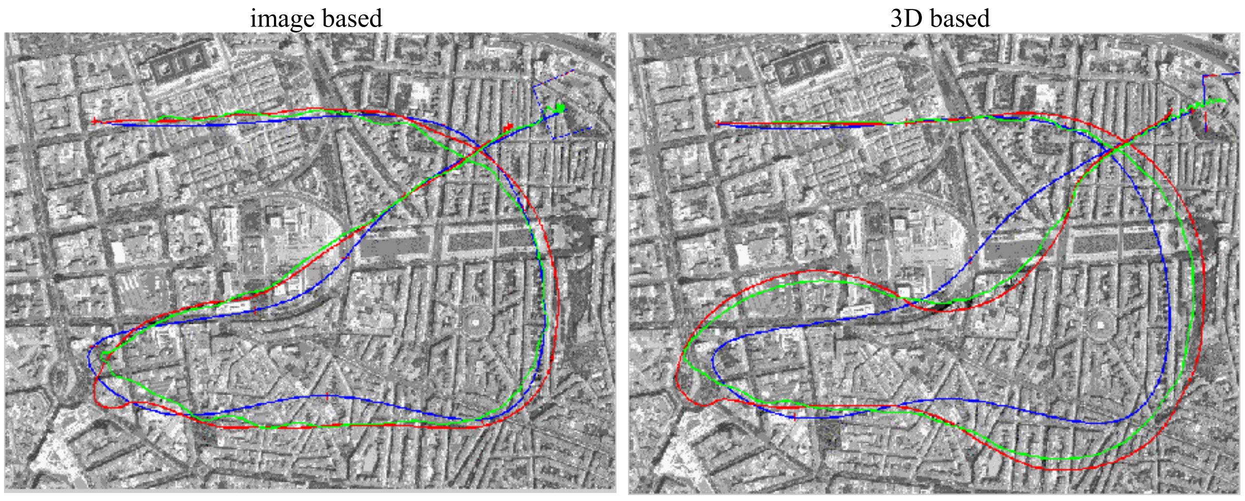


Figure 6.9 - Trajectories with perfect sensor

Both algorithms performed well, taking into account the dimension of the space occupied by the floor-map. The image based algorithm attained better performance, mainly, because in this algorithm we are using difference between pixels for the heading error unlike the 3D based algorithm that uses measurements in meters.

Note that these two algorithms aim at two different behaviors. The projection of the camera's central point in the floor plane is, in general, different from the vertical projection of the blimp's frame origin in that same plane. Due to that, in the case of the image based reference definition the blimp's frame will not end up over the end-point as in the 3D based method.

Still assuming that we have a flawless tracking and pose estimation algorithms we add an error of 10% to all extrinsic camera's parameters. The results are shown below:

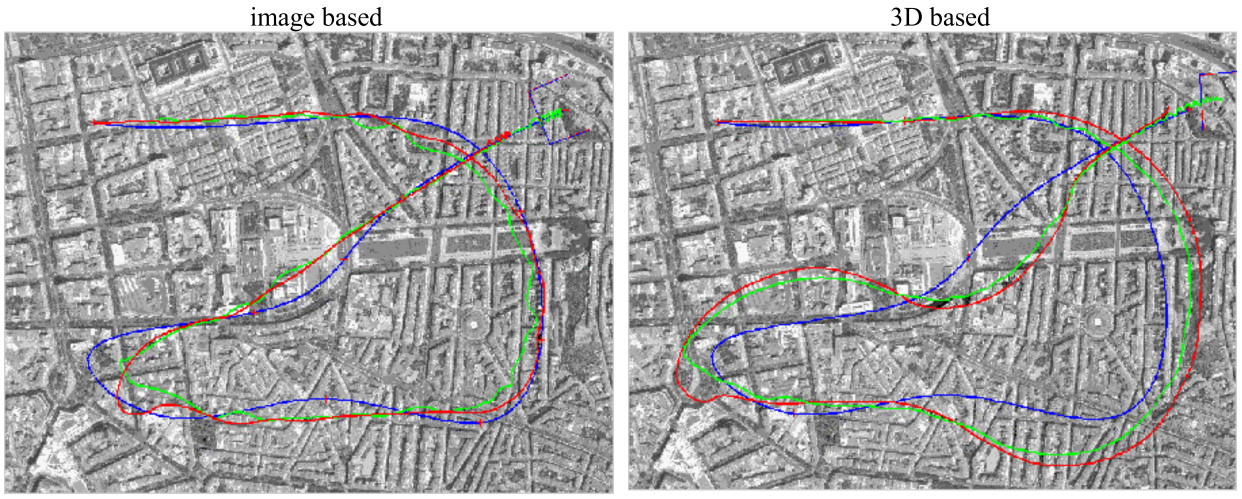


Figure 6.10 - Trajectories with 10% error in all extrinsic parameters

Comparing these two results with the previously presented ones, we see that there is little difference between the two trajectories performed by the image based algorithm. Contrarily, the performance of the 3D based algorithm clearly degrades, since it depends directly on the extrinsic parameters. In fact the minor dependence on the extrinsic parameters is the advantage of the image based algorithm. Yet, we always assumed that the camera heading is approximately equal to the blimp itself. This assumption can lead to errors if the camera is placed with large *swing* angle. However this situation is not likely to happen in ordinary situations, as the user does not desire for it to happen.

We repeated the two experiments for the image based algorithms with the tracking and pose estimation algorithm since it has the best performance as showed above. Bellow we present the resulting trajectories for 2 cases studied above:

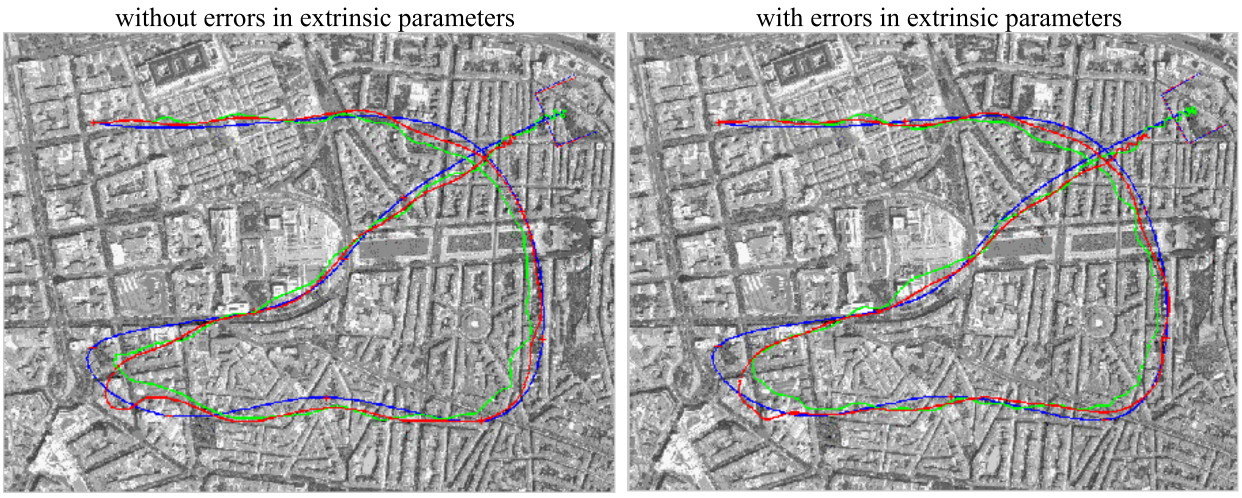

without errors in extrinsic parameters
with errors in extrinsic parameters

Figure 6.11 - Trajectories with image based error definition and similarity model

As can be observed they are similar to the obtained when assuming perfect sensor. We can conclude that, in spite of not always having the precise measurement due to accumulative errors in the tracking, the trajectory algorithm still performs reasonably. Again, we see that the algorithm is not very sensitive to error in the extrinsic parameters.

Analyzing with more detail the trajectory performed one can see that for smooth trajectories such as the first straight line, the two slaloms or the soft curve the trajectory performed is near to the desired one. However, in tighter curves and due to the inherited drift of the blimp, the distance between desired and performed trajectory rises. Yet, the algorithm tends to compensate this trajectory error and control the blimp back to the desired path.

Finally, we study the behavior of the trajectory following strategy corresponding to choosing different values for the look-ahead parameter. For the error definition algorithm we have chosen the image based one due to its better performance, and used a look-ahead value of point of 2 and 4 points. The results for this are shown in Figure 6.12.

Looking at the 2 trajectories and comparing them, some conclusions about the look-ahead parameter can be drawn.

As we can see for a look-ahead value of 2 points, the trajectory performed by the projection of the image centre is much more oscillatory since we are trying to return to the desired path in a faster way than before, leading to *roll* resulting from the Heading system's actuation. However, this behavior has a better performance when comparing it to the one shown when using a look-ahead value of 4 points.

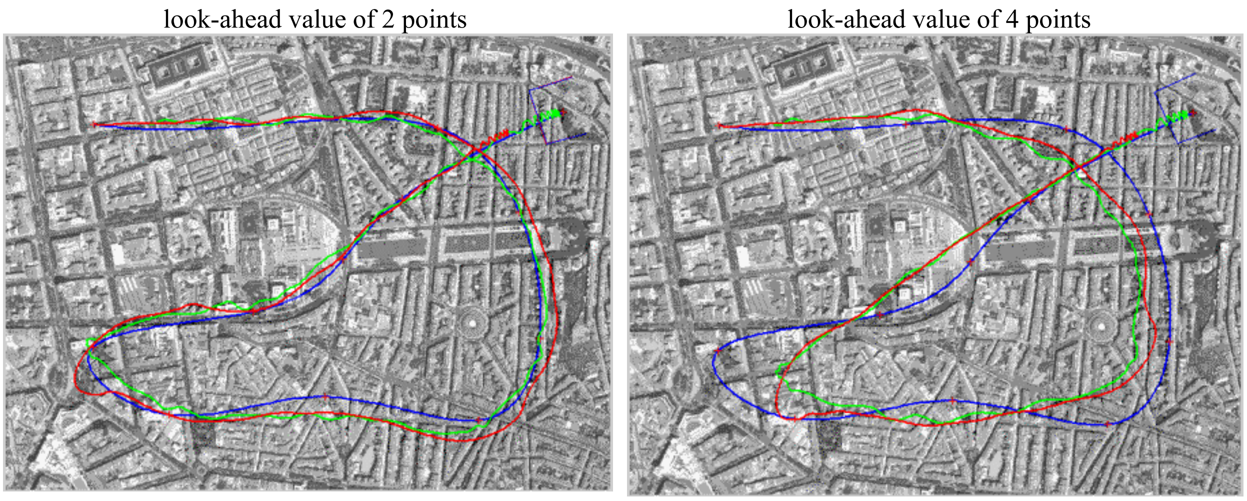


Figure 6.12 - Trajectories with look-ahead of 2 and 4 points

When setting the look-ahead, to a higher number we are effectively neglecting the current position, in relation to the desired path, and setting the desired *yaw* to a position much further. Therefore, the look-ahead parameter balances the importance of the future path planned with the intent of keeping small error between performed and the desired trajectory.

6.5. Thruster's Behavior

It is important to show some particularities of in the behaviour of the lateral thrusters and servo as this actuation system is somewhat complex.

The need to perform actuation both in the vertical direction and in the horizontal one leads to a very inconstant behaviour of the servo and thrusters, especially when null v_x velocity is demanded. In this case, as the need arises for the applied force F_x to take alternately negative and positive values, the servo is required to turn the thrusters instantly to symmetrical angle positions and the thrusters need to invert the speed of rotation.

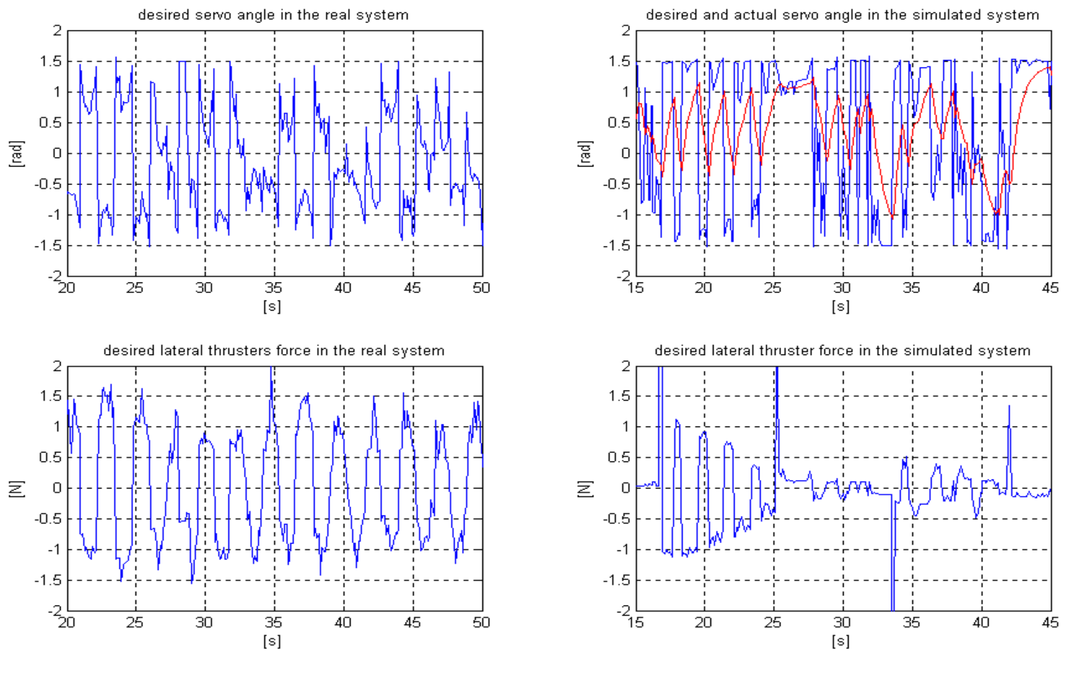


Figure 6.13 - Lateral thrusters and servo's typical desired response

In Figure 6.13, we show typical desired force values and servo angles for a station keeping manoeuvre, from both simulated and real system experiments.

The desired angle and force profiles are very oscillatory, as were expected from the observed behaviour of the blimp and errors to the references. Having the servo response available in the simulator, it is useful to notice that the real angle that the servo has is very different from the desired one, because there is not enough time for the servo angle to settle before being asked to go to a symmetrical position. This leads to a great delay in actuation, which is unmodelled in the controller's design.

The existence of these oscillatory behaviours is necessary because of the type of actuators used, having the vertical and forward force both applied by these thrusters with angle controlled by the servo. But, this behaviour, in the real system, is being demanded from a mechanical system that can suffer with this type of movements if repeated too much. The servo has to change position regularly and, worse, the thrusters have to invert rotation direction of the propellers, which leads to higher currents circulating in the motor and physical strain.

Notice, also, that the thrusters are saturated most of the time. This is the result of using motors with small power, a necessity due to the weight limitations. The stern thruster's desired force profile is similar to the ones showed for the lateral thrusters, also oscillating and operating in saturation most of the time.

7. Conclusion

In this work, the modelling and control of an aeronautical vehicle, based on vision is presented, and working solutions are proposed and tested. In Chapter 2 a full model for the robot's physics, as well as simplifications like the decoupling of the XZ and Heading systems, are presented and justified. The camera model, homography estimation routines and pose reconstruction algorithms are presented in Chapter 3. Linear controllers (LQR) and non-linear controllers (sliding mode) are proposed in Chapter 4. Trajectory and station keeping algorithms, both in 3D Cartesian space and in 2D image space are addressed in Chapter 5. Results for all the algorithms are presented in Chapter 6, both in the developed simulation environment and in the real setup. Other important contributions are described in Appendix A, where the whole parameter identification procedure for this type of system is systematized and performed. In Appendix B, a user manual for the simulator developed is presented for future use and reference.

7.1.Discussion

In this report, various methodologies for vision based control of a floating vehicle, a blimp, were proposed and tested. Pose reconstruction from homographies provides the controllers with the state variables needed to perform tracking or station keeping.

We showed that, in spite of loosing some state variables' information, it is preferable to use the similarity motion model for the homographies, instead of the planar one, because it produces less noisy estimates. This, in turn, does not affect largely the performance of the controllers, especially the non-linear ones in the XZ system as these only require the forward velocity and altitude estimates, although these state variables can be affected if the blimp gains large *pitch* or *roll* angles.

The errors in pose reconstruction for the camera, from estimated homographies, come from the estimation algorithm itself, as the necessary knowledge of the intrinsic parameters of the camera can be obtained with great precision. The vision based velocity estimation is very dependent on the distance to the floor plane. Errors in this distance disturb the velocity perception because, for the same velocity of the pixels in the camera, the vehicle can be moving faster or slower as it is nearer or further away from the floor plane. There is no way of going around this problem without the introduction of another sensor that would provide a better altitude estimate.

Another source of error comes from calibration of the extrinsic parameters (which describe the position of the camera in the blimp's frame). Deviation from the real values affects the pose estimation for the vehicle directly. This is not very serious, except in the case of different camera *swing* (i.e. rotations around the optical axis), which cannot be easily compensated for, even in purely image based control. Even so, this reconstruction phase introduces error in the loop, just before the control calculations and, therefore, affects the perception of reality by the system.

The controllers were developed for 3D Cartesian space, although some use only a few state variables, also available in the image plane. This presents a difference from other methodologies (image space controllers) where the tuning of parameters is depend on the distance to the floor and not much insight on the vehicle's dynamics. The controllers implemented are also usable in the same manner if additional sensors are added.

Problems from actuation coupling in the XZ system were dealt with in two ways. In the design stage, the LQR controller allows for the balancing of the relative weights of the X and Z actuations, which is not so intuitive in the decoupled sliding mode controllers, where this must be tuned by changing gain values directly. The servo dynamics problem can be diminished if a small dead zone is included in the lateral thrusters' actuation.

For the navigation, we have implemented two behaviors, both relying on the fact that there is a pre-defined desired path. The first behavior was based in estimating the blimp's 3D position in the world, having to use, for that, the information of the camera's extrinsic parameters. The error was defined using these 3D measurements. However, it was showed that, in this way, errors in the calibration of the extrinsic parameters largely affect the performance of the system.

Thus, a second algorithm was proposed, in which the error was measured directly in the image plane. In this manner, we were able to include the extrinsic parameters inside the control loop, thus allowing for the calibration error to be interpreted as disturbance, rather than sensor error. The objective is to drive the vehicle in a way that the target point appears in the centre of the camera image. Note, however, that in this way the navigation does not attempt to place the origin of the blimp's frame in the final point.

An offline procedure, using equidistant set of points obtained by Splines was implemented for the creation of the desired smooth path.

The simulation environment developed was an important tool in the design and analysis of the algorithms, as it allowed for the testing of various solutions and a more enlightened understanding of the functioning of all the algorithms and their specific influence in the performance of the whole loop. The dynamics behaviours in the simulator were tuned from the theoretical model and parameter identification experiments and, therefore, the simulator behaved the closest possible to the real system, for the models used.

7.2. Directions for Future Work

The wide scope of this work leaves many other possible paths not explored here, but which could much improve the results discussed previously.

Although the setup used has lift problems, it would be interesting to include some other sensors, like an altimeter, because this is one of the main variables in the system, as it largely influences the calculation of the linear velocities. Another camera could also be the solution, using stereo to recover altitude.

We showed that the planar homography model resulted in motion estimation that could be very noisy and jumpy, therefore, physically impossible. The inclusion of a Kalman Filter after the image processing and homography reconstruction should improve the results because, this way, the motion estimates could be weighted against the dynamics based prediction. This would reduce the effects of the rotation/translation ambiguity in the motion estimation of the homographies as only physically possible movements would be considered valid. As we have odometry and absolute pose sensors (inter image homography and image to floor-map homography), Kalman filtering can also be employed in merging the results into a more reliable estimate. By using these filters, the results from the planar model could be improved and, thus, surpass the results obtained with the similarity model.

As shown in our work, the servo dynamics play a very important role in the performance of the controllers. If these dynamics were included in the design of the controllers, the delay effects in the rotation of the thrusters could be predicted and compensated. This is not trivial, due to the structure of the actuators. Furthermore, the inclusion of this dynamics term would further complicate the Newton-Euler Equation with a highly non-linear term, as the dynamics are in the actuation angle.

Control in the image space, instead of the 3D space approach we used, could also be explored, although some problems should arise especially in the Heading system. We showed that it was possible to separate the dynamics of the vehicle in two and even three systems if we consider the Heading system, the X system and the Z system (these last ones controlled with non-linear controllers) and these could be used directly as image plane controllers. On the other hand, it should be interesting to analyse the result of transforming the Newton-Euler dynamics Equation into the image plane, thus loosing some state variables due to the different dimensions of these spaces but, perhaps, gaining some important insight from the new dynamics encountered.

A third controller related improvement we would suggest is not linked to the control algorithms in themselves but to the physics of the system. We have realized that the location of the centre of gravity of the vehicle has enormous importance in the oscillations observed. In tests performed in the simulator, we were able to observe that lowering the centre of gravity by a few centimetres leaded to some improvement of the blimp's behaviour. If it could be possible to shift the weight in the vehicle in order for this to happen, the behaviour of the real system would benefit largely.

Finally, the definition of a smooth curve in which to base the trajectory points was done by Splines. Another way to define this path, better in a way that it takes into account the specific behaviour of the system, is to do it using some knowledge of the blimp's physical limitations. Physics consistent path planning would take into account maximum rotation velocities, resulting lateral drifts, maximum velocities and influences of these in the vehicle's altitude value.

Appendix

A. Parameter Identification Experiments

The study of this type of airship requires some knowledge about the developed model in terms of mass, inertia and damping coefficients. It is very important to stress this, as there can be tremendously different behaviours in these vehicles depending on the values of the several parameters that compose the model.

The use of a camera as the only sensor in the system makes it essential to perform precise calibration, as the errors committed here can directly influence the performance of the system. The intrinsic parameters' calibration results are presented here and a simple method of extrinsic parameters calibration is presented as well as some results.

In what respects the blimp, first of all, we have to take into account the thrusters non-linearities that can be described by a gain function, which includes non-linearities such as dead-zone, and saturation phenomenons. The most important part of this gain function is the saturation, as this will determine the maximum force value we can apply to the vehicle.

Second, the determination of the vehicle's mass and inertia coefficients influences the modelling of its dynamic response. This can be done easily by calculation over some dimensional measurements.

Finally, in order for the model to be more accurate in terms of vehicle speed response it is necessary to obtain the damping coefficients. These coefficients determine the vehicle's maximum velocity values and, thus, are most important for an accurate model of the blimp's behaviour.

In this appendix, we describe the experiments and calculations employed in the determination of the parameters described in Chapter 2. Only in this manner, can the real blimp and actuators' model can be constructed and simulated.

A.1. Camera Calibration

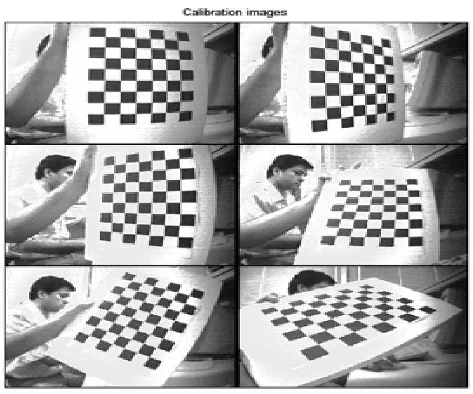
In order for the camera model presented in Section 3.2 to be used in reconstruction, it is necessary to obtain the internal and external camera parameters. The determination of these two sets of parameters is addressed in the next sections.

A.1.1. Camera Intrinsic Parameters

Since these parameters depend only on the camera's own characteristics their determination can be done independently of the experimental setup.

There are many toolboxes available for this purpose, some free to download from the internet. The toolbox we used in this work was the one by Jean-Yves Bouget and it is available at *www.vision.caltech.edu/bouguetj/calib_doc/*. This was chosen due to its operational simplicity and the tools available for the refinement and analysis of the quality of the values obtained. Also, this toolbox uses planar grids on which it detects automatically the corners used for matching. In addition, the uncertainty of the estimated parameters is given.

The set of images used for the calibration process are shown in Figure A.1. With these images we were able to obtain the intrinsic parameters for the camera as well as the radial and tangential distortion coefficients.



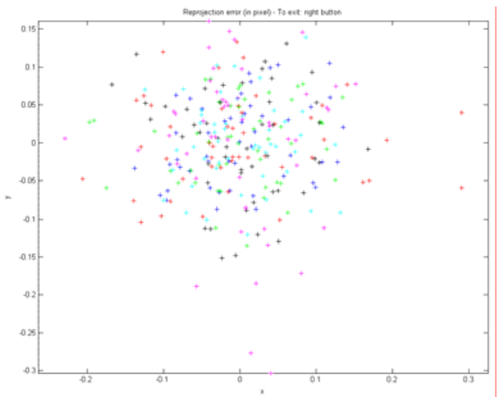


Figure A.1- Calibration images and the reprojection error

This error, here presented, is obtained when reprojecting the corner points used in calibration and comparing them to the manually defined ones. As it is possible to see we have very small, sub-pixel error and, therefore, the intrinsic parameters found are very accurate.

The calibration results for two camera resolutions are shown below:

352×288 pixels	values and uncertainties
focal length [*fx,fy*]	[332.51518 361.16843]±[2.08401 2.03027]
principal point [*cx,cy*]	[156.29349 113.94715]±[2.62849 3.38857]
skew	0
lens distortion	[-0.45529 0.17474 0.00666 -0.00135]±[0.02030 0.07676 0.00203 0.00171]

Table A.1 - Intrinsic parameters for a 352×288 pixels image resolution

192×144 pixels	values and uncertainties
focal length [*fx,fy*]	[181.29822 180.66439]±[1.39125 1.24790]
principal point [*cx,cy*]	[85.05022 57.01526]±[1.72825 2.08181]
skew	0
lens distortion	[-0.45963 0.20020 0.00615 -0.00104]±[0.02482 0.09336 0.00245 0.00209]

Table A.2 - Intrinsic parameters for a 192×144 pixels image resolution

As seen in Figure A.1 the images obtained are distorted due to the existence of the lens. Therefore, correction has to be made in order for the image to correspond to the projection models defined in Chapter 3 so it can be correctly used in the matching.

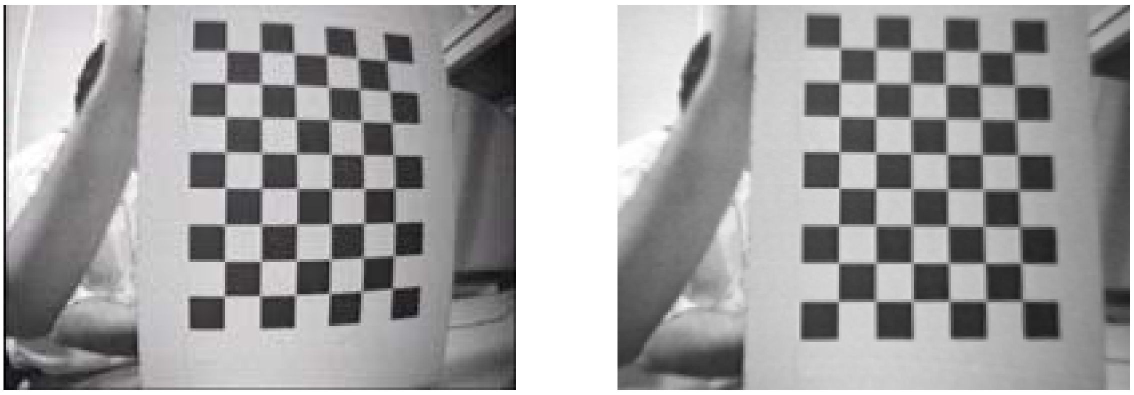


Figure A.2 - Distorted and rectified images

This correction is done in run-time, by applying a displacement mask, using the lens distortion parameters found. In Figure A.2 we show the effect of the displacement mask in one of the calibration images used.

The errors associated to the values obtained are very small and, therefore, we can conclude that the set of intrinsic parameters obtained is reliable.

A.1.2. Camera Extrinsic Parameters

As described in Figure 2.3, the blimp's frame is located in the geometric center of the envelope, assuming it has approximately an ellipsoidal shape. In Figure 3.3, the location of the camera is described in the blimp's gondola. We define the camera frame attached to the focal point of as illustrated in Figure 3.2. The extrinsic parameters define the pose of the camera in $\{b\}$. Below we describe the procedure adopted in order to obtain these parameters.

It is first necessary to calculate the homography between the camera and the world and only then, by knowing the intrinsic parameters and using the decomposition method explained in Section 3.6, we can recover the pose of the camera in the world frame. Knowing the pose of the blimp's frame in the world it is fairly easy to obtain the extrinsic parameters from the blimp to world and camera to world transformation matrixes, using the notation in [10]:

$$ {}^{b}_{c}T = {}^{w}_{b}T^{-1} \times {}^{w}_{c}T \qquad \text{(A.1)} $$

The setup devised to obtain the blimp's pose in the world frame is shown in Figure A.3. The two supports have the same height, so the blimp's *x* axis is in the horizontal position and therefore *roll* and *pitch* angles are null. Thus, the whole blimp's pose is defined by the projection of the blimp's front and back edges in the 2D floor plane and the height of the horizontal line that crosses the envelope. This height, as well as the points on the ground plane, can be easily measured, because the blimp is stationary and the supports are placed in a vertical position.

A detail for these measurements is shown also in Figure A.3. In this way, we can obtain the *x*, *y*, *z* and *yaw* for the blimp.

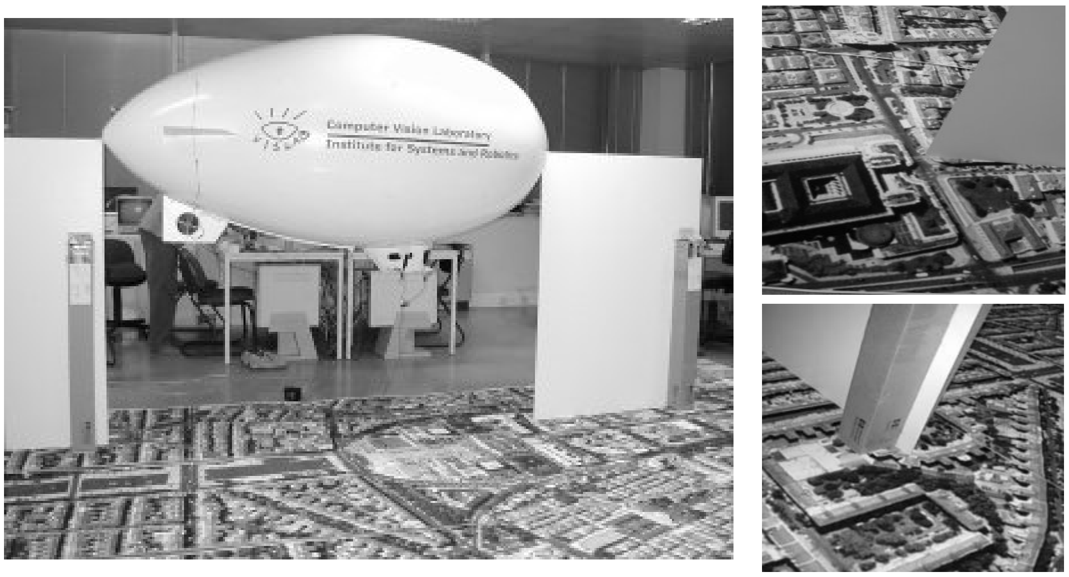

Computer Vision Laboratory
Institute for Systems and Robotics

Figure A.3 - Setup used to determine the extrinsic parameters

To obtain the image to world plane homography and, consequently, the camera pose, we use a simple method based on manually matching of points in the captured image and the floor image. The estimate is then improved with the automatic matching algorithm, which provides a larger number of correctly matched points.

Typical values for the extrinsic parameters, found through this method, are shown in the next table.

x	0.36506917221243	pan	-0.05188219492044
y	-0.01455481432092	tilt	0.03704734077319
z	0.61482343274129	swing	1.61247675014242

Table A.3 - Typical values for the pose of the camera frame $\{c\}$ in $\{b\}$

A.2. Thrusters' Position in the Blimp's Frame

Using the setup shown in Figure A.3 we were able to easily measure the exact dimensions of the blimp, as well as the location of every device that went on board. By measuring the location of the thrusters and camera in relation to the floor and knowing the location of the blimp's frame $\{b\}$, it is easy to obtain the values for the parameters introduced in Figure 2.1.

x_{stern}	-0.77
y_{stern}	0
z_{stern}	0.37
x_{star}	0.275
y_{star}	0.21
z_{star}	0.59
x_{port}	0.275
y_{port}	-0.21
z_{port}	0.59

Table A.4 - Coordinates of the thrusters in $\{b\}$ in meters

A.3. Thrusters' Non-linearities

The procedure devised to obtain the thrusters' gain profile is a relatively simple one, involving the use of a dynamometer to measure applied forces or a protractor in the case of servo angle. It is complicated to obtain a dynamometer with the required sensibility because the regular dynamometers available measure forces of the magnitude of 10N or more. This was surpassed by building our own dynamometer, to measure forces up to about 0.35N with precision. The force values, as shown in the graphs presented, are around this value.

Logical levels in the interval of [0, 255] were applied to the remote control input, through the PC interface, and for each of these, a different force or angle value of the thrusters was measured. Below, the force versus logical level graphs, are presented. It is assumed that the lateral thrusters are equal so it is only necessary to consider the force values by half if the requirements are to obtain this function for the starboard or portside thrusters alone.

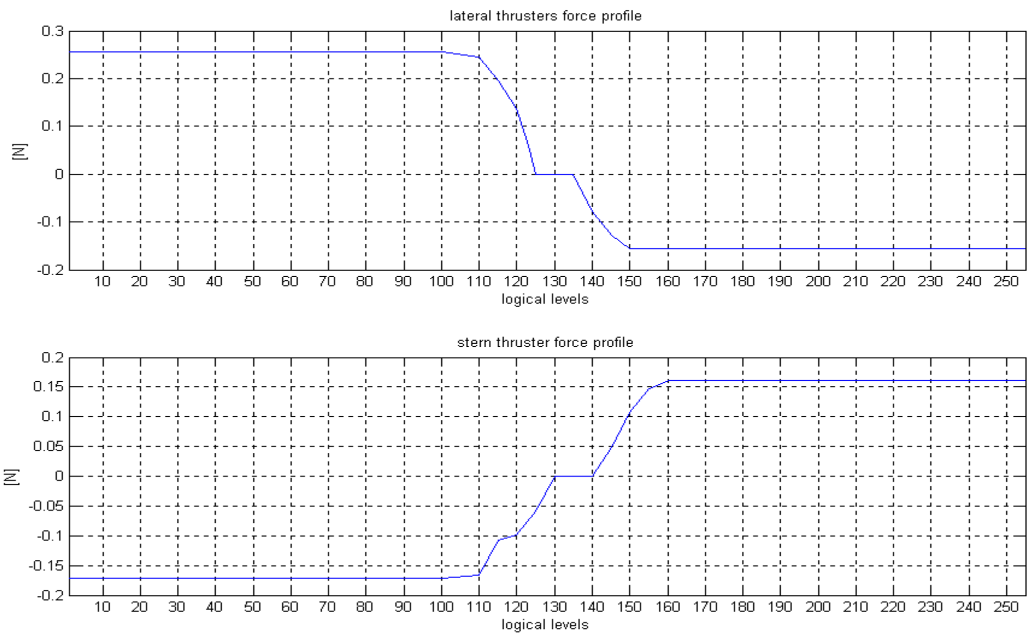


Figure A.4 - Thrusters' force profiles

From the analysis of these gain profiles, it is evident that both the stern and lateral propellers are asymmetrical. For the case of the stern propeller, this is not evident by looking at the setup but the asymmetry in the lateral ones was already expectable since the blimp is design to move forward and not backward. It is also noticeable that the real actuation levels to be used do not range from 0 to 255 but are in the intervals of [110, 150] and [110, 160] for the lateral and stern thrusters respectively, including the dead-zone that takes 10 levels in each case. These gain profiles are also dependent on the power supply of the interface box and vary a lot when the batteries loose charge.

After analysis of these results the decision was made to abandon the model determined in Section 2.3.1 because the profile encountered deviates from the quadratic form expected and thus makes it very difficult to obtain the coefficients for it. The non-linearities encountered are, thus, implemented in the simulator, by lookup tables and this knowledge is also used to try to invert their effect, in the real system. For this reason, the wake fraction phenomenon is also discarded, as it is not clear how to include it in this numerical implementation.

The servo angle versus logic levels function was obtain by means of a protractor as referred before and the setup devised for these measurements is shown in Figure A.5:

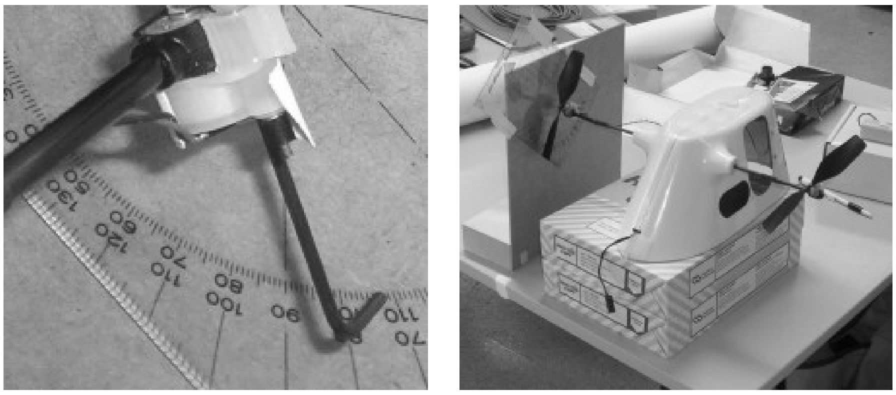


Figure A.5 - Angle measurement setup

The results obtained are presented here:

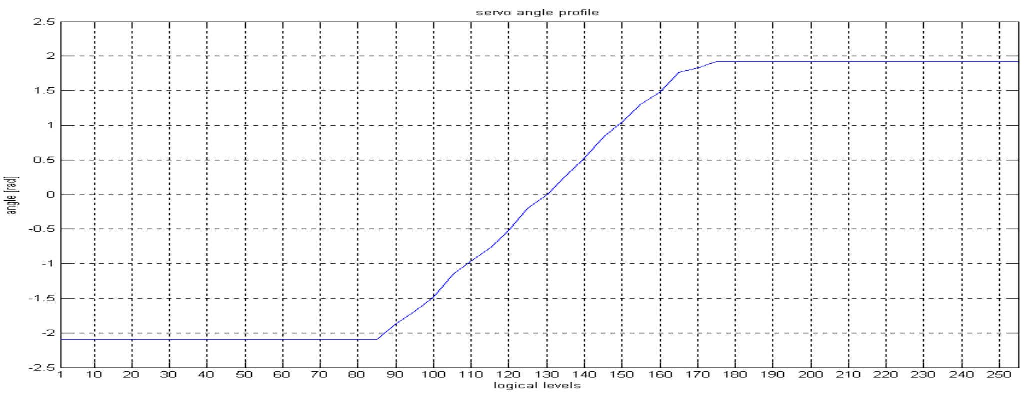


Figure A.6 - Servo angle profile

As expectable in a servo, this angle profile is linear, in spite of minor deviations from its ideal form. The zero angle position is configurable and so it is possible to alter it slightly in order for the servo response to be more symmetrical. The low resolution problem also appears in the servo, although it is of smaller magnitude (ranging the interval of [85, 175]), as the figure shows.

These experiments were, of course, affected by measurement errors. Although it is difficult to quantify these errors, it is useful to numerate their sources. The sources are:

- the radio interference related noise;
- the dynamometer calibration and force measurement;
- the battery, which must remain at a near constant charge for the whole duration (and this means performing the experiments quickly).

As for the thruster dynamics, the value for the thruster's pole was set at 10 rad/s which is considered a typical value. Having observed the servo in operation it was concluded that the pole should be at around 3 rad/s.

A.4. Mass Matrix and Inertia Coefficients

As referred in Chapter 2, it is assumed that the whole blimp can be approximated by an ellipsoid. This geometrical shape is defined by a major semi-axis with length *a*, and two symmetrical minor semi-axis with length *b*. The origin of the blimp's frame is the ellipsoid's geometrical center.

Firstly, it is necessary to determine the mass of the blimp and we did so by determining each component's mass value and then adding them up. The results for this are presented in the next table. We have used a high precision scale for weights under 350 g. The error associated with these measurements is ±0.01 g. For the envelope we use a scale with precision of ±1 g but with a larger weight limit. Below we present the weight of all the components. The total value corresponds to the gondola, the camera, one Philips 9v battery, four tail fins, one with the thruster and the blimp's envelope. In Table A.5 we present the weight for of the components as well as the uncertainty error of these measurements.

component	weight [g]
gondola with motors and batteries attached	329.09g±0.01g
camera and support rig	35.33g±0.01g
Rechargeable 9v battery (Phillips)	38.46g±0.01g
non-rechargeable 9v battery (Duracell)	46.81g±0.01g
tail fin	31.63g±0.01g
tail fin with stern engine	53.08g±0.01g
helium envelope (empty)	523g±1g
total	1.073kg

Table A.5 - Weights for the system's components

In [5] it is shown that, given these approximations, we can calculate the mass matrix parameters in Equation (2.9) using the following expressions:

$$\begin{cases} a_{11} = \dfrac{\alpha_0}{2-\alpha_0} m \\ a_{22} = a_{33} = \dfrac{\beta_0}{2-\beta_0} m \\ a_{44} = 0 \\ a_{55} = a_{66} = \dfrac{1}{5} \dfrac{(b^2-a^2)^2(\alpha_0-\beta_0)}{2(b^2-a^2)+(b^2+a^2)(\beta_0-\alpha_0)} m \end{cases}, \begin{cases} \alpha_0 = \dfrac{2(1-e^{2)}}{e^3}\left(\dfrac{1}{2}\ln\left(\dfrac{1+e}{1-e}\right)-e\right) \\ \beta_0 = \dfrac{1}{e^2} - \dfrac{1-e^2}{2e^3}\ln\left(\dfrac{1+e}{1-e}\right) \\ e = 1-\left(\dfrac{b}{a}\right)^2 \\ m = \dfrac{4}{3}\pi\rho a b^2 \end{cases} \tag{A.2}$$

$$I = \int_V \rho(x,y,z) \begin{bmatrix} y^2+z^2 & -xy & -xz \\ -xy & z^2+x^2 & -yx \\ -xz & -yz & x^2+y^2 \end{bmatrix} dxdydz \Rightarrow \begin{cases} I_{yy} = I_{zz} = \dfrac{4}{15}\pi\,\rho a b^2(a^2+b^2) \\ I_{xx} = \dfrac{8}{15}\pi\,\rho a b^4 \end{cases} \tag{A.3}$$

The values for these parameters are shown in Table A.6.

a_{11}	0.29846 kg
a_{22}	0.70180 kg
a_{33}	0.70180 kg
a_{44}	0.00000 kg
a_{55}	0.04767 kg
a_{66}	0.04767 kg
I_{xx}	0.1710 kg
I_{yy}	0.3341 kg
I_{zz}	0.3341 kg

Table A.6 - Values for the parameters in the mass and inertia matrix

The location of the centre of gravity was found through observation of an experiment where the blimp was left free and without actuation, thus acquiring its natural pitch angle. The centre of gravity is located in the vertical direction below the centre of buoyancy. The relative distance was found by observing the *roll* behaviour of the blimp and tuning the simulated vehicle's characteristics accordingly. The values found in this way are x_G=-0.03 m, z_G=0.25 m and y_G=0 m.

A.5.Damping Coefficients

With the limited tools and space available at the lab, there was the need to devise a way to find the values for the model's damping coefficients. The experiments done rely on the fact that these damping coefficients control the values of the maximum speeds and therefore, assuming some decoupling, can be found approximately by analysis of the speed profiles. The speed profiles necessary were obtained through online processing in the Viscon software mentioned in Section 6.1, which provides real-time estimates of the speed, and then confirmed by offline processing of the sequential images filmed by the camera.

Let us consider the linearized model presented in Equation (2.5). If we eliminate the couplings from the Coriolis matrix and the effect of external forces like gravity or buoyancy we have:

$$\begin{cases} F_x = (m + a_{11})\dot{v}_x + mz_G\dot{w}_y + D_{vx}v_x + D_{vxvx}v_x\left|v_x\right| \\ F_y = (m + a_{22})\dot{v}_y - mz_G\dot{w}_x + mx_G\dot{w}_z + D_{vy}v_y + D_{vyvy}v_y\left|v_y\right| \\ F_z = (m + a_{33})\dot{v}_z - mx_G\dot{w}_y + D_{vz}v_z + D_{vzvz}v_z\left|v_z\right| \\ N_x = (I_{xx} + a_{44})\times\dot{w}_x - mz_G\dot{v}_y + D_{wx}w_x + D_{wxwx}w_x\left|w_x\right| \\ N_y = (I_{yy} + a_{55})\dot{w}_y + mz_G\dot{v}_x - mx_G\dot{v}_z + D_{wy}w_y + D_{wywy}w_y\left|w_y\right| \\ N_z = (I_{zz} + a_{66})\dot{w}_z + mx_G\dot{v}_y + D_{wz}w_z + D_{wzwz}w_z\left|w_z\right| \end{cases} \tag{A.4}$$

Considering that the vehicle has attained steady state for all velocities, i.e. these will be constant, the Equations in (A.1) can be further simplified.

$$\begin{cases} F_x = D_{vx}v_x + D_{vxvx}v_x\left|v_x\right| \\ F_y = D_{vy}v_y + D_{vyvy}v_y\left|v_y\right| \\ F_z = D_{vz}v_z + D_{vzvz}v_z\left|v_z\right| \\ N_x = D_{wx}w_x + D_{wxwx}w_x\left|w_x\right| \\ N_y = D_{wy}w_y + D_{wywy}w_y\left|w_y\right| \\ N_z = D_{wz}w_z + D_{wzwz}w_z\left|w_z\right| \end{cases} \tag{A.5}$$

These equations found are perfectly valid if we now consider that we are working in discrete time. It is now important to note that it is not possible to attain steady state velocities for all of them, it is only possible to perform the experienced devised for the linear velocities and the w_z rotation velocity. Because the equations found are all similar, the following deduction is only presented for the v_x velocity, as it is only a matter of replacing the variable's names to obtain the other desired results.

If the vehicle is moving at low speed, these equations allow us to express the damping as a function of velocity and force applied.

$$D_{vx}v_x \simeq F_x \Leftrightarrow D_{vx} \simeq \frac{F_x}{v_x} \tag{A.6}$$

Now that we have the linear damping coefficient, we can calculate the quadratic one:

$$D_{vxvx} = \frac{F_x}{v_x |v_x|} - \frac{D_{vx}}{|v_x|} \tag{A.7}$$

In the limited space available at the ISR, it was impossible to run the v_z velocity experiment and it is also probable that the couplings here discarded would have too much influence so this experiment was not performed. As for the v_y and w_z velocities experiments, we could not do them in an isolated manner, as there is only one thruster (the stern thruster) that allows for the actuation in these. Therefore, the corresponding damping coefficients had to be extracted from coupled experiments in these two velocities, as detailed in the following sections.

A.5.1. Determination of D_{vx} and D_{vxvx}

To obtain these coefficients we accelerated the blimp to two different linear velocity values, low speed and high speed, by applying two different logic values to the lateral thrusters. The velocity profiles obtained from processing the images filmed with camera are presented in Figure A.7.

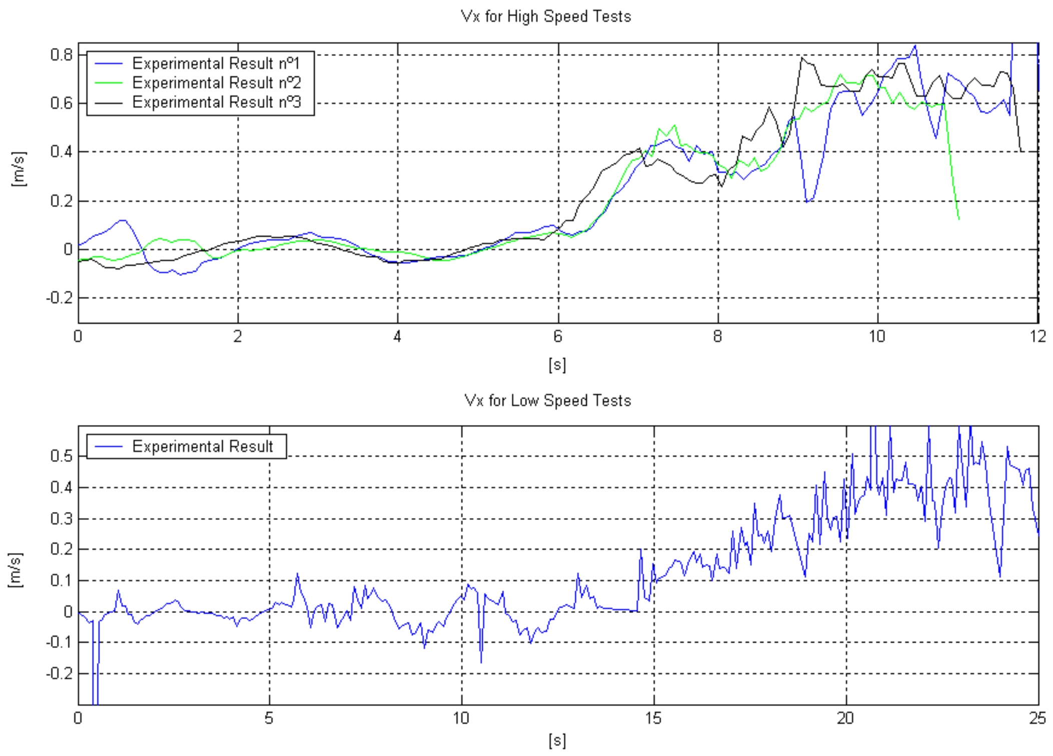


Figure A.7 - Velocity profiles for v_x

Although we barely had room to reach steady state velocity with saturated motors, it is clear, from the two high-speed experiments that it was reached.

The velocity profiles obtained are not very smooth. However, it is still possible, by means of mere visualisation, to determine the values of the steady-state velocities. Moreover, the peaks that appear in v_x profile are due to two facts:

- the filmed images were strongly blurred and noisy, therefore the image matching algorithm looses accuracy;
- the velocity update from consecutive poses tends to accumulate error. This error is zeroed periodically, by the image to mosaic matching, thus changing the position estimate abruptly.

The first three experiments were performed by applying a logic level of 10, which leads to maximum force being applied. In the last experiment, the logic level applied was 123.

We found v_x to be 0.65 m/s when the blimp is driven to maximum speed and 0.27 m/s when it is at low speed. Below, the resulting calculations for the D_{vx} and D_{vxvx} are presented.

$$D_{vx} \simeq \frac{F_x}{v_x} \Rightarrow D_{vx} \simeq \frac{0.0784}{0.27} = 0.2904N/ms^{-1} \tag{A.8}$$

$$D_{vxvx} = \frac{F_x}{v_x|v_x|} - \frac{D_{vx}}{|v_x|} \Rightarrow D_{vxvx} = \frac{0.2548}{(0.65)^2} - \frac{0.2904}{0.65} = 0.1563N/m^2s^{-2} \tag{A.9}$$

A.5.2. Determination of D_{wz}, D_{wzwz}, D_{vy} and D_{vyvy}

Again, we accelerated the blimp to two different rotational velocity values, low speed and high speed, by applying two different logic values to the stern thruster. For the low and high velocities, we have applied logical values of 143 and 170 respectively. The velocity profiles obtained from processing the images filmed with camera are presented in Figure A.8 and Figure A.9.

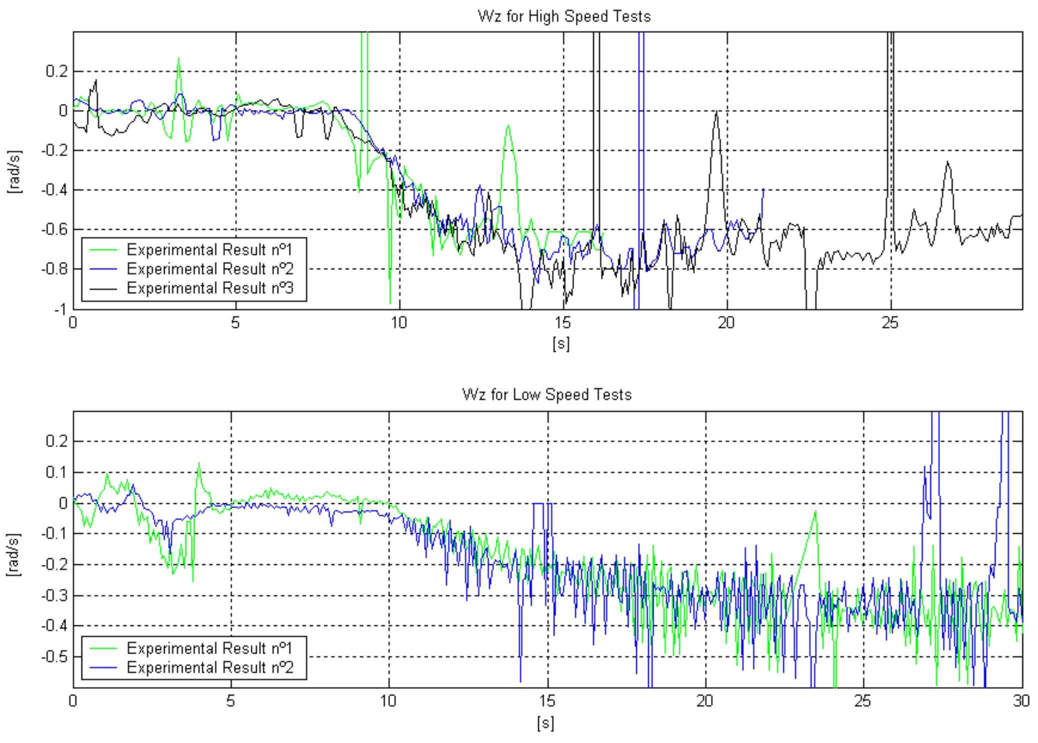


Figure A.8 - Velocity profiles for w_z

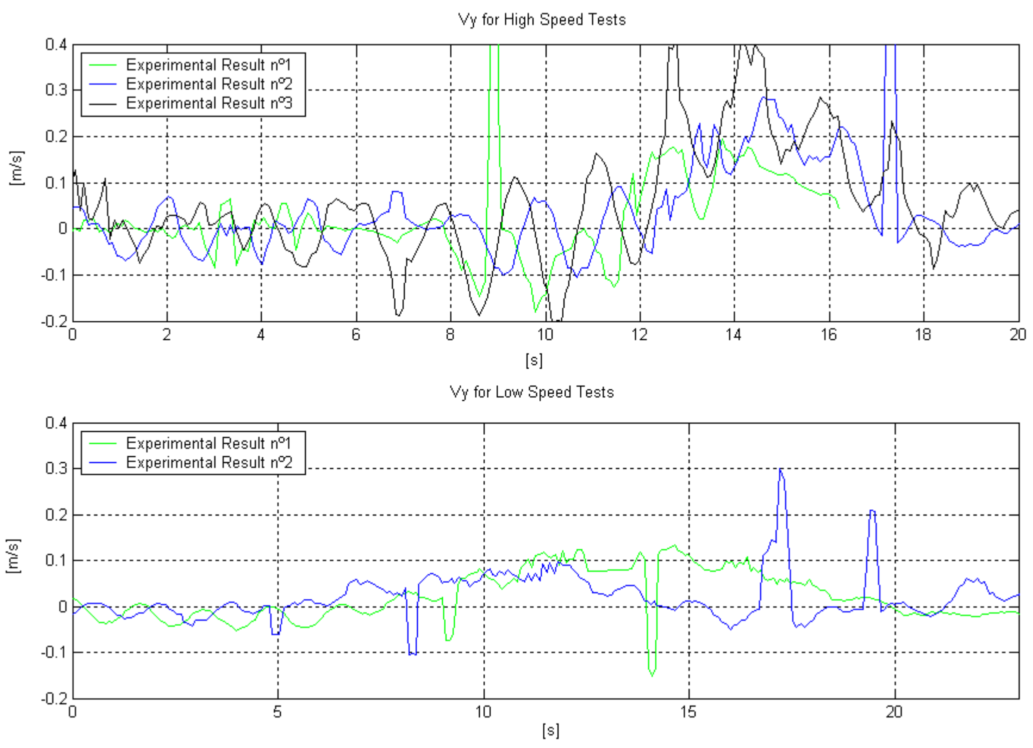


Figure A.9 - Velocity profiles for v_y

Again, several peaks appear in the velocity profiles of w_z as a result of the facts referred in the previous section but, also due to the *yaw* angle being limited to [$-\pi$, π]. When this angle crossed the border from π to $-\pi$ or vice-versa a peak would occur. As for the v_y, note that steady state velocity is not really obtained and therefore the calculations made on these results can be used just as an initial estimate to be tuned.

We found w_z to be 0.3 rad/s when the blimp is driven to maximal speed and 0.68 m/s in low speed. As for v_y, high speed is considered to be at 0.16 m/s and low speed at 0.21 m/s.

Below, calculations for D_{wz}, D_{wzwz}, D_{vy} and D_{vyvy} are presented. Note that $N_z=F_y x_{stern}$.

$$D_{wz} \simeq \frac{F_y x_{stern}}{w_z} \Rightarrow D_{wz} \simeq \frac{0.049 \times 0.77}{0.30} = 0.1258 N/ms^{-1} \tag{A.10}$$

$$D_{wzwz} = \frac{F_y x_{stern}}{w_z |w_z|} - \frac{D_{wz}}{|w_z|} \Rightarrow D_{wzwz} = \frac{0.1666 \times 0.77}{(0.68)^2} - \frac{0.1258}{0.68} \simeq 0.0924 N/m^2 s^{-2} \tag{A.11}$$

$$D_{vy} \simeq \frac{F_y}{v_y} \Rightarrow D_{vy} \simeq \frac{0.049}{0.05} = 0.98 N/ms^{-1} \tag{A.12}$$

$$D_{vyvy} = \frac{F_y}{v_y |v_y|} - \frac{D_{vy}}{|v_y|} \Rightarrow D_{vyvy} = \frac{0.1666}{(0.15)^2} - \frac{0.98}{0.15} = 0.8711 N/m^2 s^{-2} \tag{A.13}$$

A.5.3. Final Determination of the Damping Coefficients

Having the simulation environment available (see Appendix B), we tuned the values found in order for the simulation output to be more similar to the real velocity profiles found in these parameter identification experiments.

We started by simulating the experiments for the v_x profiles. For the low and high speed experiments a force of 0.0784 N was applied. After tuning D_{vx} a force of 0.1563 N was then applied to the blimp so that we could tune the high speed behaviour. We found that D_{vx}=0.1904 and D_{vxvx}=0.1763. Below, in Figure A.10, we compare the simulated behaviour with the experimental one.

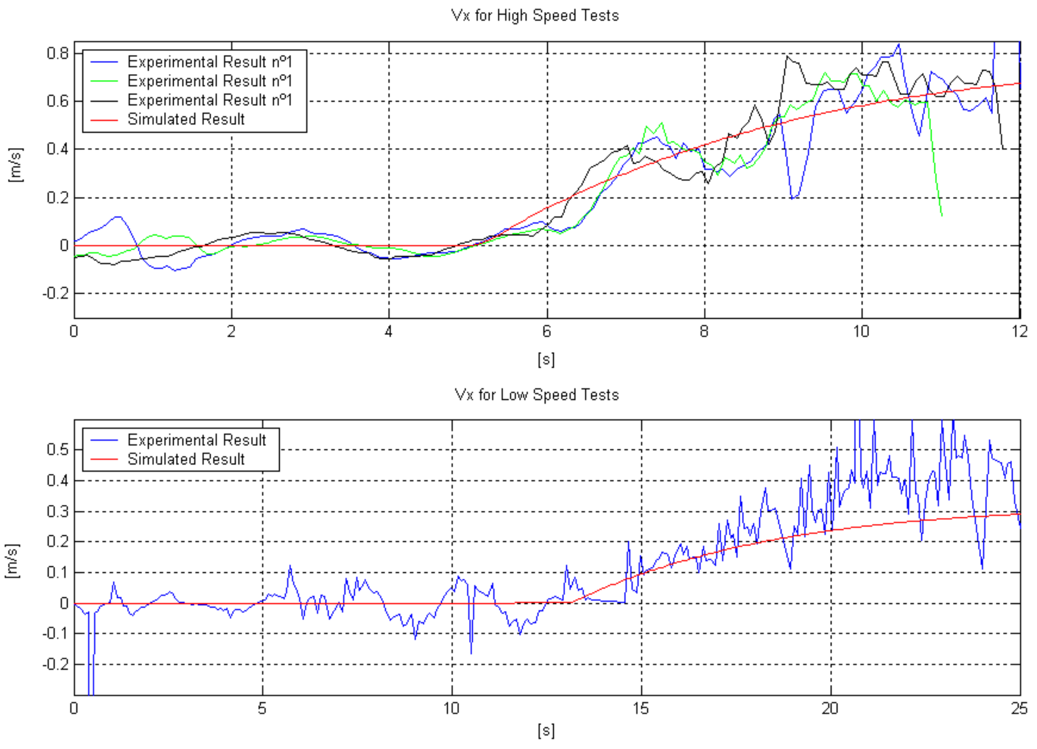


Figure A.10 - Profile comparison between real and simulated v_x velocity

The resulting linear coefficient is much smaller and the quadratic one is higher. This could have happened if the force applied during the real experiment was higher than we expected.

For the damping coefficients in z and y, we simulated an actuation force of 0.049 N for the low speed tests and 0.1666 N for the high speed tests. Below, in Figure A.11 and Figure A.12, we present the resulting profiles for w_z and v_y:

For w_z profiles, the coefficients found resulted in a dynamic response very similar to the one obtained in the experimental results without much tuning, therefore the coefficients found are correct and quite accurate.

For the v_y profiles, the values were tuned so that they could be as similar to the experimental profiles as possible. The exact same oscillations could not be achieved but this may not b desirable. In this experiment, the v_y velocity is not quite constant and unmodelled couplings with *roll* and *yaw* exist, which certainly disturb the estimation of pose.

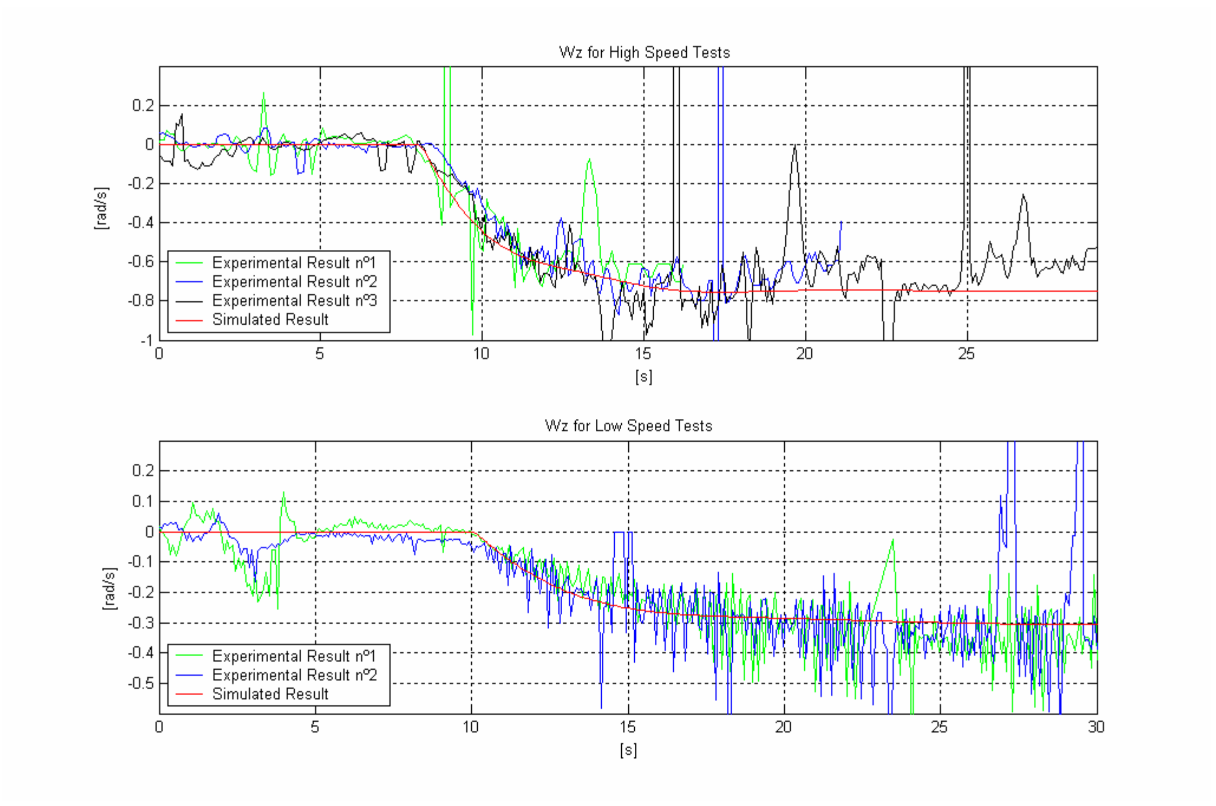


Figure A.11 - Profile comparison between real and simulated w_z velocity

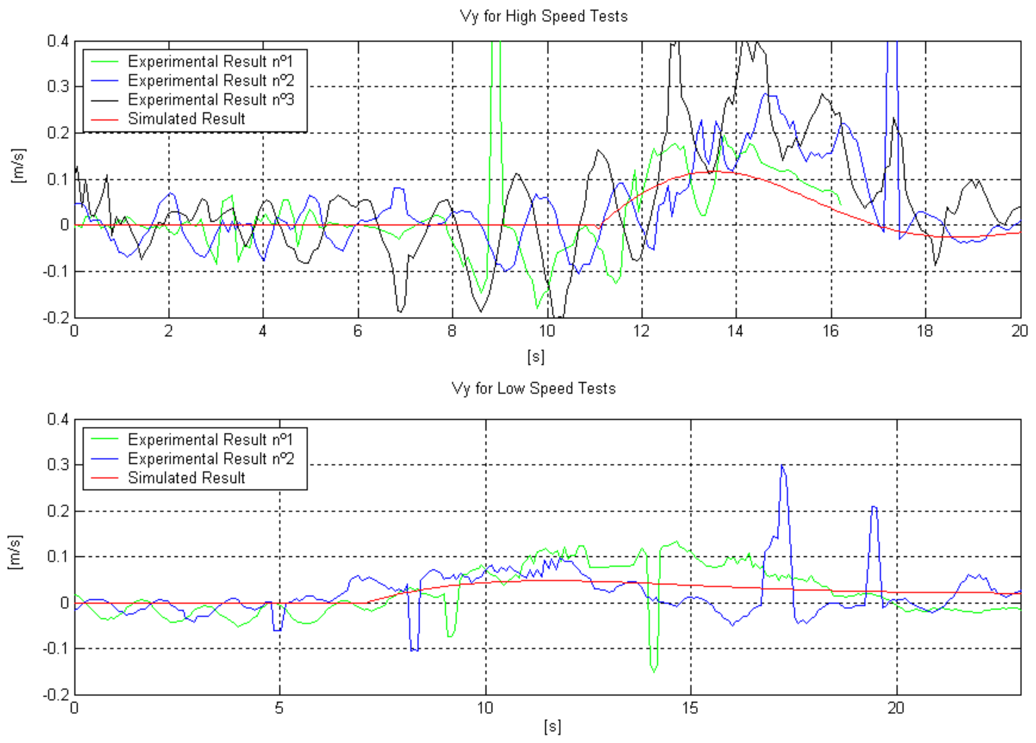


Figure A.12 - Profile comparison between real and simulated v_y velocity

As for D_{vz} and D_{vzvz}, due to the ellipsoidal shape of the blimp, despite the problems found in their determination, we assume that the damping coefficients are equal to D_{vy} and D_{vyvy} as the best guess. For the remained coefficients, since there is no way they could be found by similar experiments, we have tuned them simultaneously with the others as they exerted some minor influence in the dynamics behaviours we intended to achieve.

Thus, the final values, tuned in the simulation environment, are presented in the following table.

D_{vx}	0.1904	D_{vxvx}	0.1763
D_{vy}	0.78	D_{vyvy}	0.8011
D_{vz}	0.78	D_{vzvz}	0.8011
D_{wx}	0.1	D_{wxwx}	0.01
D_{wy}	0.3	D_{wywy}	0.1
D_{wz}	0.09	D_{wzwz}	0.0954

Table A.7 - Damping values used in the simulator

B. *BlimpSim* User Guide

In this appendix, we present a complete description of the simulator developed in the scope of this project. The *BlimpSim* simulator, which is briefly presented in Chapter 6 and more detailed here, is based on some previous simulation work done by Sjoerd van der Zwaan and Nuno Ricardo Gracias for their Masters and PhD thesis [1] and [2].

This simulation environment was built in order to implement and study the various algorithms presented in our report, but it also allows further developments thus providing a manageable and useful test bed for future work.

B.1. Introduction

The *BlimpSim* is an aeronautics control simulator, implementing the model of a generic blimp with actuation thrusters and both control and trajectory algorithms. It also simulates image capturing from an onboard micro camera, and subsequent image processing for state estimation, in parallel with the physics simulation.

It is implemented in *MatLab*®*6.5/Simulink*®*5.0* with its main simulation interface being the S*imulink*'s user interface, although internal processing and configuration files are simple *MatLab Editor* files. *Simulink* was chosen due to the ease of implementation of simple or more complex dynamic models as well as including complex user-defined functions for the image processing part. This development environment also provides the user with intuitive signal output analysis tools, which can easily be employed in the building of sophisticated Graphical User Interfaces (GUI). The structure of the simulator is presented in Figure B.1

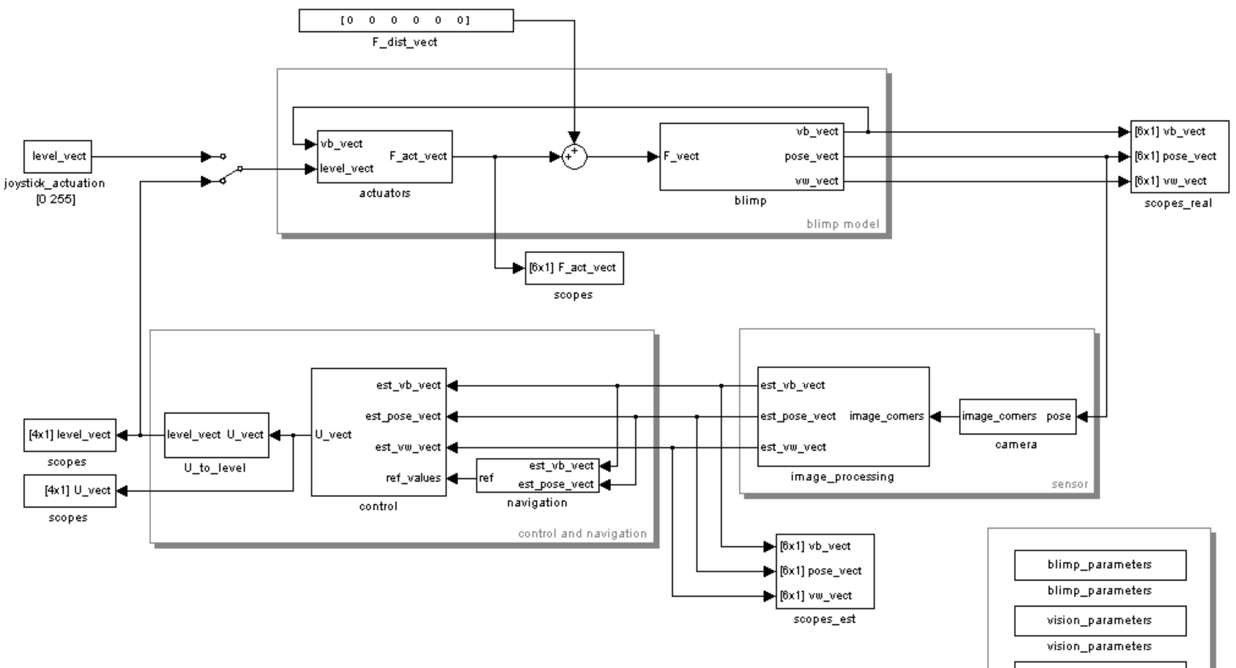


Figure B.1 - The *BlimpSim* simulator's main structure

As for the simulation files, the main file (*blimpsim_mdl.mdl*) is the one that directs the simulation, calling other functions and algorithms, whose files are kept in the *./mdlfunctions*, *./imgmatch*, *./extras* and *./guifunctions* directories. The main file implements the block's structure of the simulator and appears in Figure B.1 as seen in the *Simulink* environment.

The non-standard functions built for *Simulink* execution are *s-functions*, which are called in execution order, as any other *m-function* or standard block, but present an advantage over *m-functions* in that they are pre-compiled, making the simulation flow much faster.

Modularity is a concern that is ever present in the simulator structure as this allows a user that is not that familiar with its internal functioning to fully use the simulator. On the other hand, this makes it possible for someone who desires to change, for example, the actuators structure or the blimp's parameters, the image processing routines or the camera specifications, to do it easily by just substituting or altering the corresponding blocks of the simulator without having to go into the details of the other blocks.

Having this in mind, this guide is divided into two main Sections, the first addressing those users who are only interested in using the simulator as it is, without interest in learning about its internal workings, and the second for others that wish to introduce major changes in the simulation or simply alter some of the original routines.

B.2. The Simulator User Interface

An objective of this simulator is to be easily usable and configurable. With this objective in mind, an intuitive interface was developed for simple and easy access to the simulation control variables and system parameters.

There are three blocks that, through user-defined parameters, control the simulation environment initialization and flow (the *vision_parameters*, the *blimp_parameters*, and the *control_parameters* blocks). In addition to these, if it is desired to do some extra processing over the system variables, there is an empty *s-function* block just for that purpose, called *extra_processing*, placed in the *./extras* directory. Moreover, the *extra_parameters* routine, placed in that same directory, has access to the vision parameters and images, in the initialization stage, and allows the user to add and initialize his own data fields in the *userdata* structure of the *vision_parameters* block. For more details on this, see Section B.3.

B.2.1. Configuration Files

The parameters that describe the blimp's physical characteristics, the controller weight matrixes or gains, the vision processing algorithms' parameters, and other simulation flow controls, can be supplied by the user in separate files.

Three default files are provided with the simulator and they are *default_blimp.m*, *default_control.m*, and *default_vision.m* in the *./mdlfunctions* directory. These just need to be copied and renamed and then they can be altered at will. Explicit and abundant comments guide the user through the variables present in these files.

There is the need, however, to change one line in the simulator's *s-functions* (*blimp_parameters*, *control_parameters* and *vision_parameters* blocks), which is the line that defines the configuration files' names (`ud.source='default_control';` in the case of the control parameters block, for example).

B.2.2. Signal and Graphical Outputs

The usual *Simulink scopes* blocks are present for the main system variables like the real and estimated pose and velocities, the actuation values or the real force values applied to the blimp. Many other scopes blocks are available inside the main blocks represented in the simulator structure depicted in Figure B.1.

Vision related outputs are of a graphical nature and are not easy to display using *Simulink* standard blocks. Therefore, it was necessary to create a graphical output that shows the camera's image or the floor plane and the projection of the image's borders in this plane. Thus, two figure windows are available and used for this. The filmed image is updated in the *camera* block, which is described in greater detail in Section B.3.4 and the image borders are also updated by the *camera* block in the case of the real borders (in blue) and by the *tracking* functions in the *vision_processing* block in the case of the estimated image borders (in red).

It is also in this floor-map image display that trajectory points are chosen by the user.

B.2.3. User Defined Processing

For user defined purposes, the simulator includes two functions, one for parameter configuration and the other for execution in the loop, synchronous with as any other simulator block. These functions are located in the *./extras* directory.

The *extra_parameters* function is a normal *MatLab m-file* function, which is called by the *vision_parameters* block and, having access to the entire block's initialized *userdata* structure, can add fields to this. Pay attention not to change system variables as this can alter and even invalidate the simulation.

The *extra_processing* block implements an *s-function* that has access to the same *userdata*, but in runtime, and therefore can perform user algorithms on top of the simulation environment variables, keeping the results in the user defined fields initialized in *extra_parameters*.

B.3. The Simulator Core Structure

In order to fully understand some options taken in the design of the simulator it is useful to be familiar with the *Simulink* environment and the way *s-functions* work.

The next Sections address more specific aspects of each block implemented and corresponding processing but some previous explanations are necessary to facilitate the complete understanding of the whole simulation structure.

First, note that, there are only three blocks with actions defined for the initialization phase of the simulation. This was done because it allows us to concentrate the simulator definitions in three configuration files that relate directly to initialization blocks. This option also makes it easier to provide the simulator with a GUI to alter, even in runtime, the definition of the simulation by just accessing these blocks' *userdata* structures.

Secondly, some function blocks output values received in their input, without even changing them. This happens, as it is the only way to preserve the desired block execution sequence, because of the way *Simulink* determines this. Data flow is therefore sequential, as desired, but passing through the vision block's *userdata*.

Finally, care was taken to use intuitive variable names along the code and explicit function headers in order to help the user understand the implemented function or algorithm.

B.3.1. The Parameters' Blocks

These three blocks are the ones that allow the user to define a series of parameters going from the blimp's model to the control gains or the choice of vision routines, controllers, etc. The values for these parameters are kept, during runtime, in each block's *userdata* structure in order for it to be accessible to all the other simulator blocks and future GUIs.

The *blimp_parameters* block defines all the kinematical and dynamics properties of the blimp, as measured and calculated from various tests performed on the real setup (see appendix A). They are all one-dimensional constants.

`ud.` `gui_active` – for the GUI to be able to control the block
`source` – name of the definitions file to use
`x_cg` – x coordinate of the centre of gravity
`z_cg` – x coordinate of the centre of gravity
`Ixx` – inertia coefficient
`Iyy` – inertia coefficient
`Izz` – inertia coefficient
`Ixz` – inertia coefficient
`a11` – added mass term
`a22` – added mass term
`a33` – added mass term
`a44` – added mass term
`a55` – added mass term
`a66` – added mass term
`Dvx` – linear damping coefficient
`Dvy` – linear damping coefficient
`Dvz` – linear damping coefficient
`Dwx` – linear damping coefficient
`Dwy` – linear damping coefficient
`Dwz` – linear damping coefficient
`Dvxvx`– quadratic damping coefficient
`Dvyvy` – quadratic damping coefficient
`Dvzvz` – quadratic damping coefficient
`Dwxwx` – quadratic damping coefficient
`Dwywy` – quadratic damping coefficient
`Dwzwz` – quadratic damping coefficient
`g` – gravity's acceleration
`m` – blimp's mass
`x_port` – x coordinate of the portside thruster's location
`y_port` – y coordinate of the portside thruster's location
`z_port` – z coordinate of the portside thruster's location
`x_star` – x coordinate of the starboard thruster's location
`y_star` – y coordinate of the starboard thruster's location
`z_star` – z coordinate of the starboard thruster's location
`x_stern` – x coordinate of the stern thruster's location
`y_stern` – y coordinate of the stern thruster's location
`z_stern` – z coordinate of the stern thruster's location

Table B.1 - Block's u*serdata* structure for the *blimp_parameters* block

The *control_parameters* block is somewhat different in that it performs all the necessary calculations for the controllers design, subsequently initializing all the necessary blocks that implement the controllers. This is chosen so that it becomes possible to tweak the performance of the control system without having to leave the simulation environment to perform the necessary calculations. In the beginning of every simulation, the calculations are performed according to the specifications presented in the control parameters block. The next table shows the various fields in this structure. Most of these variables' values are used in gain blocks in the control subsystems. The values found for the weight matrixes and some sliding mode gains are presented in Chapter 6.

`ud.` `gui_active` – for the GUI to be able to control the block
`source` – name of the definitions file to use
`Q_dir` – state variables' weight matrix for the Heading system [5×5]
`R_dir` – actuation weight constant for the Heading system
`K_lqr_dir` – LQR gains for the Heading system [1×5]
`p_d` – desired poles for the Heading s-mode controller [1×4]
`K_sm_state_dir` – gains for the linear part of the Heading s-mode [5×1]
`h_sm_state_dir` – eigenvector of the system's matrix for the Heading s-mode [5×1]
`inv_b_sm_state_dir` – inverse gain for the Heading system's s-mode
`Q_xz` – state variables' weight matrix for the XZ system [6×6]
`R_xz` – actuation weight matrix for the XZ system [2×2]
`Ki_lqr_xz` – LQR integrator gain matrix for the XZ system [2×2]
`K_lqr_xz` – LQR gains for the XZ system [2×4]
`lambda_vx_int` – integrator error gain for the X system s-mode
`lambda_vx` – velocity error gain for the X system s-mode
`thickness_vx` – boundary layer thickness for the X system s-mode
`eta_vx` – convergence gain for the X system s-mode
`Kd_vx` – differential gain for the X system s-mode
`lambda_z` – position error gain for the Z system s-mode
`lambda_vz` – velocity error gain for the Z system s-mode
`lambda_z_int` – integrator error gain for the Z system s-mode
`thickness_z` – boundary layer thickness for the Z system s-mode
`eta_z` – convergence gain for the Z system s-mode
`Kd_z` – differential gain for the X system s-mode

Table B.2 - Block's u*serdata* structure for the *control_parameters* block

The *vision_parameters* block defines the geometrical parameters of the camera sensor, initializes the vision processing system and finally keeps all intermediate results, updated during runtime, at every simulation step. The block's *userdata* structure contains the images filmed by the camera, real and estimated homographies, estimated pose and velocities and other variables related to the image processing like the floor-map image, initial position of the blimp and trajectory points in the world frame $\{w\}$. The next table shows the various fields in this structure:

`ud.` `gui_active` – for the GUI to be able to control the block
`source` – name of the definitions file to use
`sampletime` – sample time of the control and image processing loop
`showmode` – weather to show or not the images filmed and the floor image-map
`floormap_img` –image containing the map of the floor
`dx` – horizontal resolution of the image
`dy` – vertical resolution of the image

`save_imgs` – signal weather to save the image sequence in a predefined directory
`results_dir` – path to the images saved
`img_num` – index of the current image
`img_width` – with of the camera filmed image
`img_height` – height of the camera filmed image
`K_real` – real intrinsic parameters of the camera [3×3]
`T_cb_real` – real extrinsic parameters of the camera [4×4]
`K_calib` – calibrated intrinsic parameters of the camera [3×3]
`T_cb_calib` – calibrated extrinsic parameters of the camera [4×4]
`init_pose_real` – real initial pose of the blimp [1×6]
`H_wc0_real` – real homography from world plane to camera plane [3×3]
`H_c0c_real` – real inter image homography [3×3]
`real_c` – real camera plane corners projected onto the world plane [2×4]
`line_real_h` – handle of the corners' line printed in the floor-map window
`init_img` – initial image taken from the camera
`curr_img` – current image taken by the camera
`prev_img` – previous image taken by the camera
`trajectory_x` – x coordinates of the desired trajectory points
`trajectory_y` – y coordinates of the desired trajectory points
`floor_h` – handle to the floor-map figure window
`spline_num` – number of points, in the Spline trajectory, between desired trajectory points
`spline_break` – minimal separation between consecutive Spline points
`ind_ref` – index of the currently desired destiny point
`ind_act` – index of the current point
`H_c0c_est` – estimated inter-image homography [3×3]
`floor_h` – handle to the camera figure window
`H_wc0_est` – estimated homography from world plane to camera plane [3×3]
`init_pose_est` – estimated initial pose [1×6]
`pose_vect_est` – current estimated pose [1×6]
`vb_vect_est` – current estimated blimp velocity in $\{b\}$ [1×6]
`vw_vect_est` – current estimated blimp velocity in $\{w\}$ [1×6]
`est_c` – estimated camera plane corners projected onto the world plane [2×4]
`cam_centre_point` – vector with the image's central point projected onto the floor plane
`line_est_h` – handle of the estimated corners' line printed in the floormap window

Table B.3 - Block's u*serdata* structure for the *control_parameters* block

B.3.2. Implementation of the Blimp's Model

The blimp's physical model is implemented in the block called *blimp*, shown here.

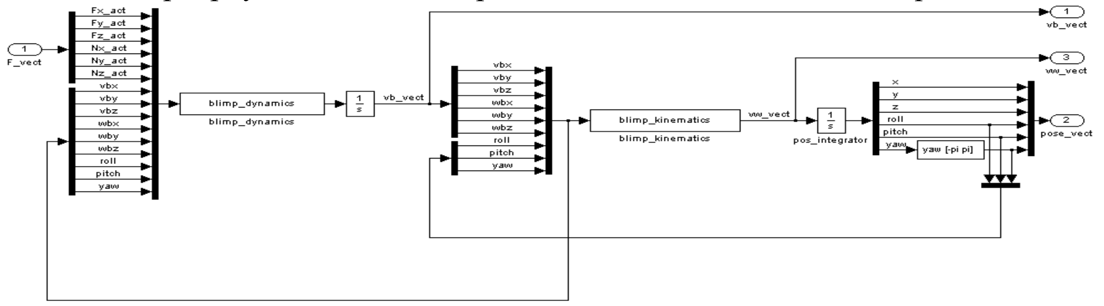


Figure B.2 - The internal structure of the *blimp* block

The vehicle kinematics and dynamics presented in 2.2 are here implemented in several *s-function* blocks. These calculate the accelerations resulting from the applied forces (using the Newton-Euler formula from Equation (2.5)) and performing the jacobian transformation in (2.4) to world coordinates.

Note that the output angles are generally small, except for the *yaw*, which can assume any value and is therefore artificially limited to [-*π* ,*π*] in the subsystem shown at the *yaw* output. Applied forces come in vector *F_vect* and the blimp's velocity and pose are outputted in *vb_vect*, *vw_vect* and *pose_vect* (velocity in {*w*} and in {*b*} and pose in {*w*} respectively).

The *s-functions* are in the *blimp_dynamics.m* and *blimp_kinematics.m* files placed in the *./mdlfunctions* directory. Note that these blocks do not have an initialization run and load the values for the blimp's constants (initialised according to the user definitions) from the *blimp_parameters* block, at each simulation step.

B.3.3. Implementation of the Actuator's Model

It was decided to place the actuators' simulation in a different block from the blimp as this makes it possible to change actuators configuration easily and without affecting other parts of the simulator. This enables one to test, for example, new configurations for the thrusters, asymmetries in the lateral thrusters or even slower or faster motor's response times.

The next figure shows the inner structure of the *actuators* block that is already presented in the structure presented in Figure B.1. Again, the separation was made between the dynamics behaviour and the kinematics of the thrusters' system.

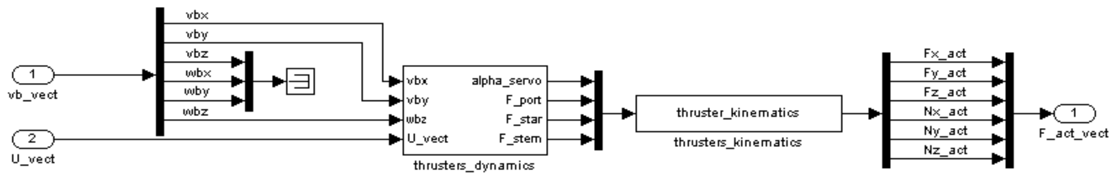


Figure B.3 - The internal structure of the *actuators* block

The *thruster_kinematics* block, at each simulation step, obtains the location of the thrusters in the blimp's reference frame {*b*} (see Figure 2.4) and calculates the resulting forces and momentums applied to the blimp's body. This is implemented in the *thrusters_kinematics.m* file in the *./mdlfunctions* directory.

The dynamics block's composing blocks are shown in Figure B.4.

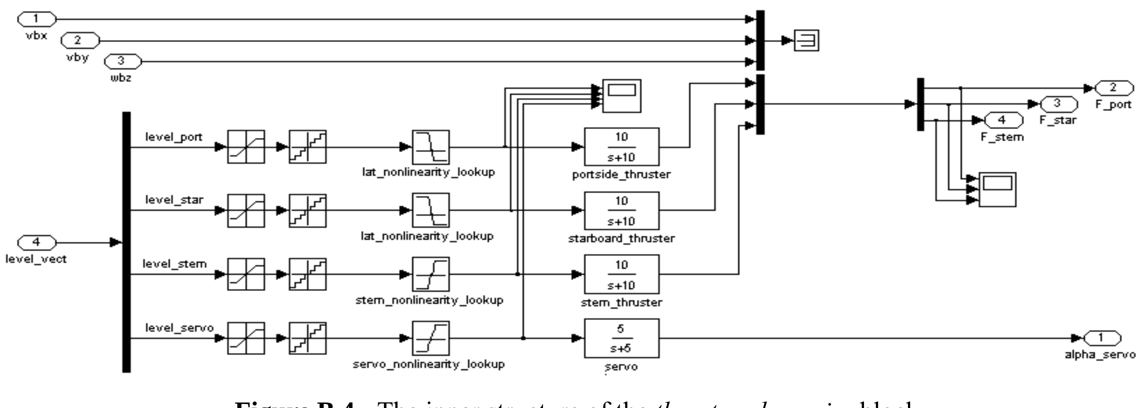


Figure B.4 - The inner structure of the *thruster_dynamics* block

As shown, the non-linearities in the thrusters are implemented by means of lookup tables built on the values determined experimentally by the means described in Appendix A.

Despite the fact that the wake fraction phenomenon is not implemented, for the reasons explained in Section A.3, the variables, in theory, necessary for the simulating of its effect, are accessible in the block for future implementation of this to be facilitated.

Note that the thrusters' input comes in logical levels (in the interval of [0, 255]) as is the case in the real system, where this is the way actuation is performed from the PC.

B.3.4. Implementation of the Camera's Model

The camera is the first block in the control loop and therefore it was decided that this is were the sampling would be simulated, by means of a Zero Order Hold (ZOH) block. It is also here that the processing delay from the image and control algorithms' calculations is simulated, just at the output of the camera. This is shown in Figure B.5.

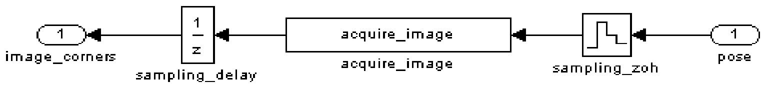


Figure B.5 - The inner structure of the *camera* block

An *s-function* block composes the main block, which simulates the capture of the filmed frames, according to the camera's extrinsic and intrinsic parameters, as described and modelled in Chapter 3 and provided by the user. The camera parameters are loaded from the *vision_parameters* block at each simulation step and, by using these and the current pose of the camera, the filmed image is built (without interpolation so that the process is faster).

Note that radial distortion from camera lens was not modelled nor implemented, although it is an easy task to include it in the simulator routines once the algorithm is done.

The main file, *acquire_image*, and subsequent function files that contain these routines are in the *./mdlfunctions* directory. In the main file, it is easy to add functionalities such as variable image noise, radial distortion, lighting changes or even loss of frames.

As was said before, there is the need to output some data in order for *Simulink* to correctly define the simulation order of the blocks. The four corners of the image plane projected in the ground plane were chosen but any other output can be sent as long as coherence is maintained in the subsequent functions.

B.3.5. Implementation of the Image Processing Routines

The image processing routines are divided into the two main blocks that correspond to the two main processing algorithms which need to be implemented: the estimation of homographies based on image features (tracking) and the reconstruction of the blimp's movement based in those homographies (pose estimation). Both these function blocks are included in the *image_processing* block in Figure B.1 that we show here in more detail.

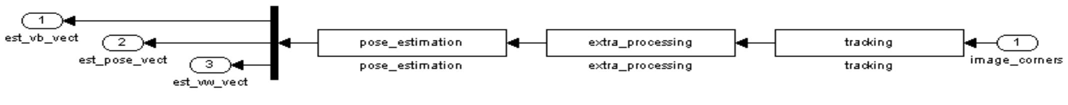


Figure B.6 - The inner structure of the *image_processing* block

As explained in B.2.3 the *extra_processing* block is to be used for user defined processing algorithms and therefore this Section will not address it. The blocks that are here explained are the *tracking* and *pose_estimation* blocks, both implemented in *s-functions* which are kept in the *./mdlfunctions* directory.

B.3.5.1. Tracking

This block performs the image matching procedures in order to obtain the homographies that describe the relation between the sequential camera images or the camera image and the available ground plane image.

At every iteration step, the block produces an estimate of the inter-image homography and updates the last image to floor-map homography. This is done by using the last known homography multiplied by the currently estimated one. At the end of a series of iteration steps (configurable value), the image to floor-map matching algorithm is activated and the true homography is updated, thus eliminating accumulated position error, as explained in 3.5.

It is here that the routines developed in [1] are included in the simulator, being called from this block. These matching and homography estimation routines are placed in the *./imgmatch* directory and more profound configurations should be made there.

Again, note that this block receives as input the corners of the images in the ground plane and outputs them without using them for the processing. This guarantees the preservation of the desired execution order.

B.3.5.2. Pose Estimation

The pose estimation routines described in Section 3.6 are in this block. This includes pose and velocities reconstruction from image to floor-map or from inter image homographies. The world velocities are calculated either from the transformation of the blimp's velocities in $\{b\}$ to the world frame $\{w\}$ trough the jacobian defined in Equation (2.4) or from the displacement calculated from successive world frame poses.

As outputs, this block makes available the pose of the blimps as well as the velocities in the blimp's frame $\{b\}$ and in the world frame $\{w\}$.

Again, note that this block receives as input the corners of the image in the ground plane in order to preserve the desired execution order.

B.3.6. Implementation of the Control Algorithms

The control algorithms are implemented in three main blocks shown here in Figure B.7 and they are the decoupled XZ and Heading systems and the trajectory block. All the controllers are presented in Section 4.2 and 4.3, except for the PID controllers, included just for comparison purposes.

The choice of controller is done through the *vision_parameters* block by setting the two constant blocks in the figure to the value defined by the user in the configuration files, which then controls the enabling of the desired controllers.

Figure B.7 presents the inner structure of the block where the control algorithms are implemented.

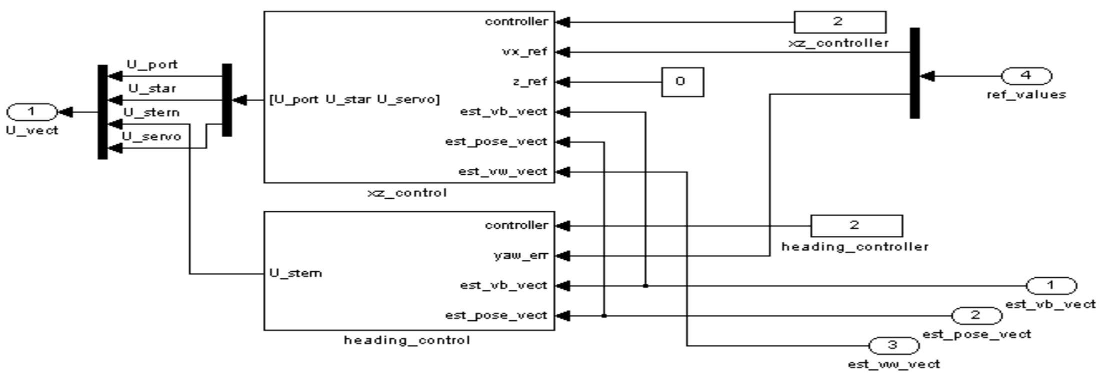


Figure B.7 - The inner structure of the *control* block

Both controller blocks have a similar internal functioning. The state vector is composed of the state variables required for each model and these are fed to each controller as shown in the next figures. If it is necessary to add more controllers, they can be placed here. There is only the need to change the enabling mechanism to support another controller.

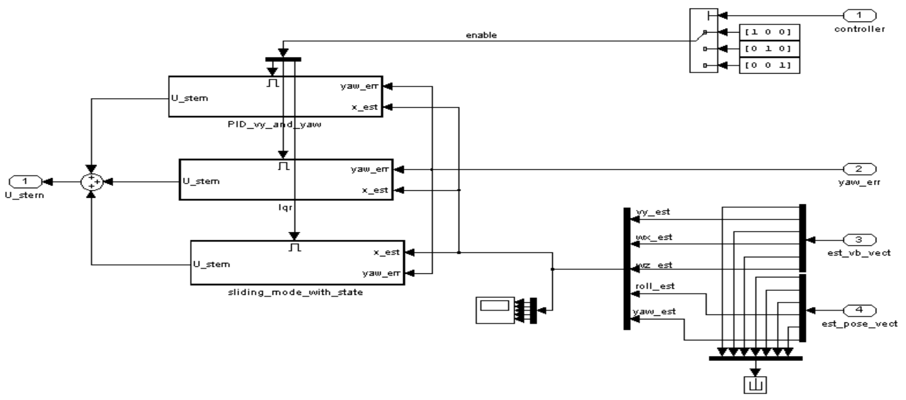


Figure B.8 - The inner structure of the *xz_control* block

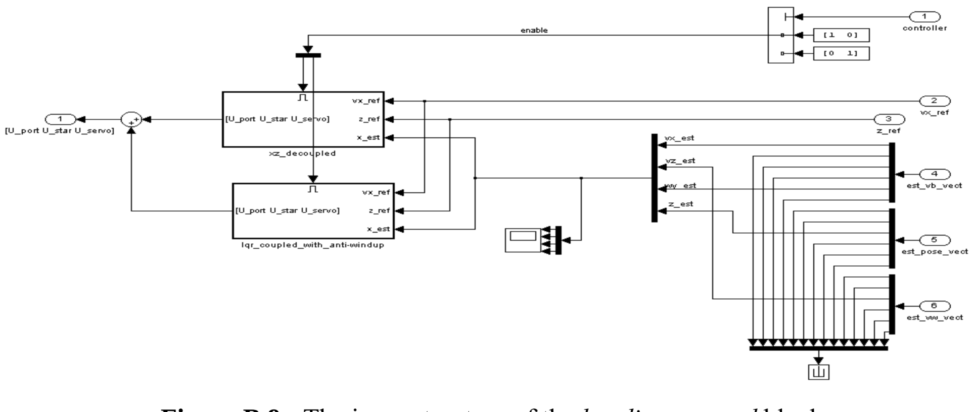


Figure B.9 - The inner structure of the *heading_control* block

The controller blocks are placed in parallel but are equipped with an enable switch controlled by the constant blocks. Therefore, only the calculations for the chosen controller are done, allowing for a faster simulation.

Various *Simulink scope* blocks are present inside in order to allow for the analysis of the performance of each controller.

Most of the controller gains can be defined directly in the parameters initialisation but it was chosen that for the LQR the gain matrixes should be calculated in the initialisation step from the weight matrixes Q and R defined by the user. This makes it more intuitive to define the controllers and any change in the characteristics of the blimp is contemplated in the new controllers calculated.

B.3.7. Implementation of the Navigation Algorithms

The trajectory following algorithms described in Section 5.1 are implemented here and it was chosen that this block should output the desired v_x velocity values and the *yaw* error for calculation simplicity. The navigation was implemented in an *s-function* in which the error can be defined using the 3D based or image based algorithms.

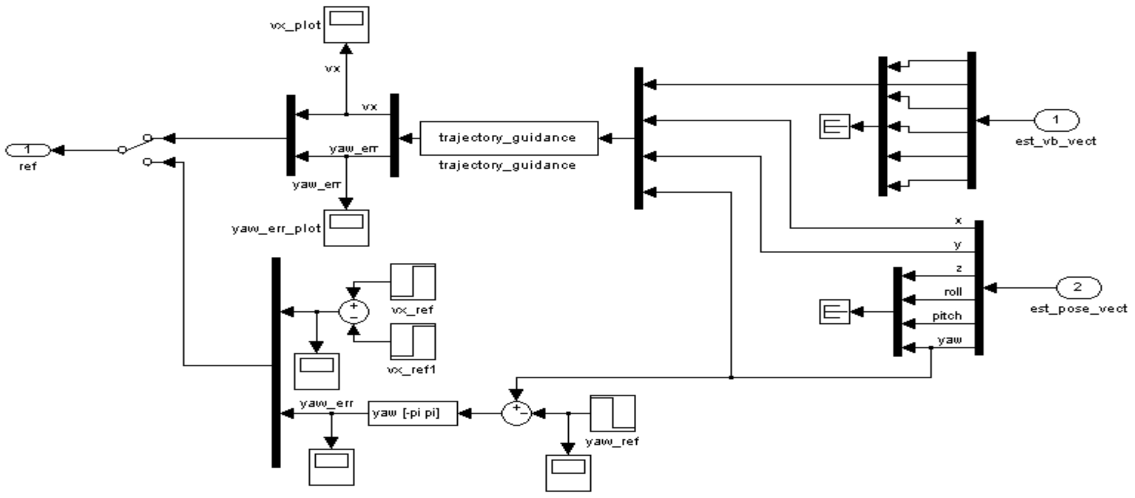


Figure B.10 - The inner structure of the *navigation* block

The pre-defined path can be inputted in one of two ways, chosen by the user through, a flag implemented in the configuration file. If this string is put to '*user*', the points are inputted manually in a GUI and the number of input points can be chosen. If, on the other hand, the string is put to '*defined*' the trajectory points used are the ones that are pre-defined in the configuration file, vectors *trajectory_x* and *trajectory_y*, in metric coordinates.

Finally, the look-ahead parameter can only be updated inside the function that produces the *yaw* and v_x error. For the case of image based the file to be updated is named *trajectory_image.m* and for 3D based the file is to be update is named *trajectory_3D.m*.

C. Controller Gains and Weight Matrices

In this appendix, we present the values used for the controller gains and weight matrixes introduced in Chapter 4 and some step response tests showing the final behaviour of the system, with the image processing algorithms in the loop. The values presented here were found experimentally through testing in the simulator and tuning, until a satisfactory response for the system was found. The image processing algorithms run at approximately 12 Hz with floor-map matching every 4.2 seconds.

The weight matrixes are built so that the main state variables in the system are privileged against the others. This is the case of the *yaw* in the Heading system and the v_x velocity and the z altitude in the XZ system.

C.1. Heading System

We applied a sequence of two desired *yaw* steps to the Heading system in order to test the response to small and large angle errors.

For the LQR controller the weight matrixes and the resulting gains used are:

$$\begin{cases} Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 14 \end{bmatrix} \\ R = 4 \end{cases} \Rightarrow K \begin{bmatrix} -0.0695 \\ -0.23656 \\ -1.4889 \\ -0.67759 \\ -1.8708 \end{bmatrix}^T \qquad \text{(C.1)}$$

For the sliding mode controller the desired poles, resulting gain matrix and non-linear controller parameters are:

$$\begin{cases} p_{desired} = \begin{bmatrix} -0.4 + j4.5 \\ -0.4 - j4.5 \\ -0.4 \\ -0.3 \\ 0 \end{bmatrix} \Rightarrow K = \begin{bmatrix} -0.05979 \\ 0.07050 \\ -0.02867 \\ -0.06900 \\ -0.06900 \\ 0 \end{bmatrix}^T \\ k = 3 \\ \sigma = 1.7 \end{cases} \qquad \text{(C.2)}$$

The results are showed graphically in Figure C.1.

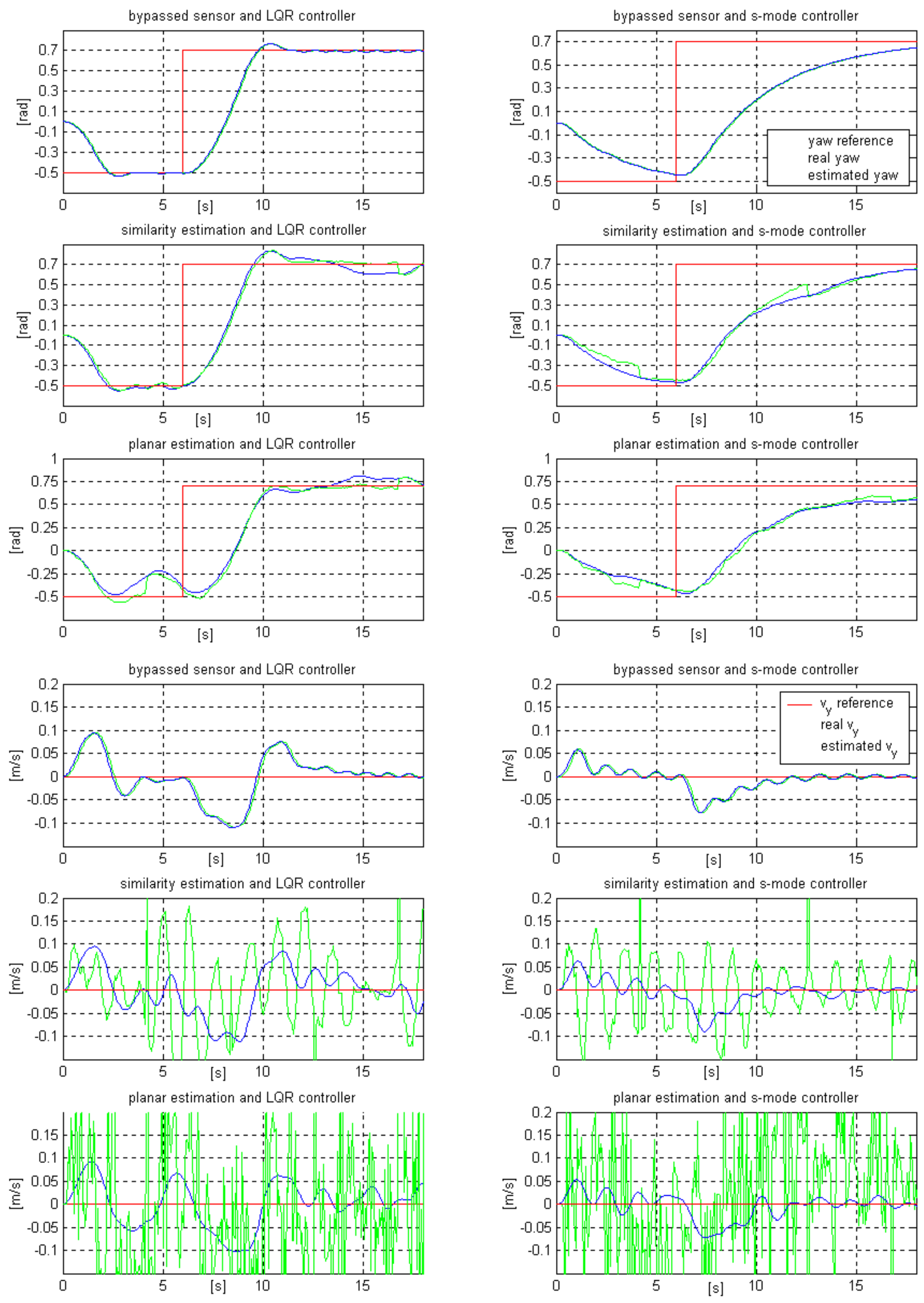


Figure C.1 - System response for the Heading controllers

There is a great difference in the response of the system for the LQR controller and the sliding mode controller, especially in *yaw*, the most important state variable to control in this system. The sliding mode controller presents a very bad step response difficult to improve due to the tuning of this controller not being very intuitive. Tuning of the non-linear controller parameters lead only to chattering or oscillation. The specification of the desired poles obeyed only the criterion of not placing the too far from the original ones in order not to have a permanently saturated desired output force.

The absence of *roll* estimates in the case of the similarity model does not make much difference in the convergence of the LQR controlled system, as the oscillation in *roll* is reflected in the error in v_y. Thus, the perceived v_y is larger than the real one from the influence of the lateral displacement of the pixels in the image in the similarity model.

Note that the image processing, using the similarity model, provides very good estimates of the *yaw* angle as would be expected.

It is not shown here, but, when rotating, the blimp also gains great backwards drift, even reaching velocities in the vicinity of 0.15 m/s. This behaviour denotes that rotating the blimp excites the couplings between the Heading and XZ systems.

C.2. XZ System

For the XZ system, we applied a series of desired v_x steps and, as for z, the objective of the control is to maintain constant height, equal to the estimated initial value.

For the coupled LQR controller the matrixes are the following:

$$\begin{cases} Q=\begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \\ R=\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \end{cases} \Rightarrow \begin{cases} K=\begin{bmatrix} 3.55665 & 0.20036 \\ 0.64483 & 4.12669 \\ 0.34470 & 0.12217 \\ 0.72022 & 5.86951 \end{bmatrix}^T \\ K_I=\begin{bmatrix} K_{Iv} & K_{Iz} \end{bmatrix}=\begin{bmatrix} 3.13818 & 0.27550 \\ -0.55010 & 3.13818 \end{bmatrix} \end{cases} \tag{C.2}$$

The non-linear controllers' gains used four the decoupled X system and Z system are presented in (C.3) and (C.4) respectively:

$$\begin{cases} k = 6 \\ \sigma = 3.33 \\ K_d = 0 \\ \lambda = 1 \end{cases} \tag{C.3}$$

$$\begin{cases} k = 6 \\ \sigma = 2 \\ K_d = 0.9 \\ \lambda = 1 \end{cases} \tag{C.3}$$

The results are showed graphically in Figure C.2.

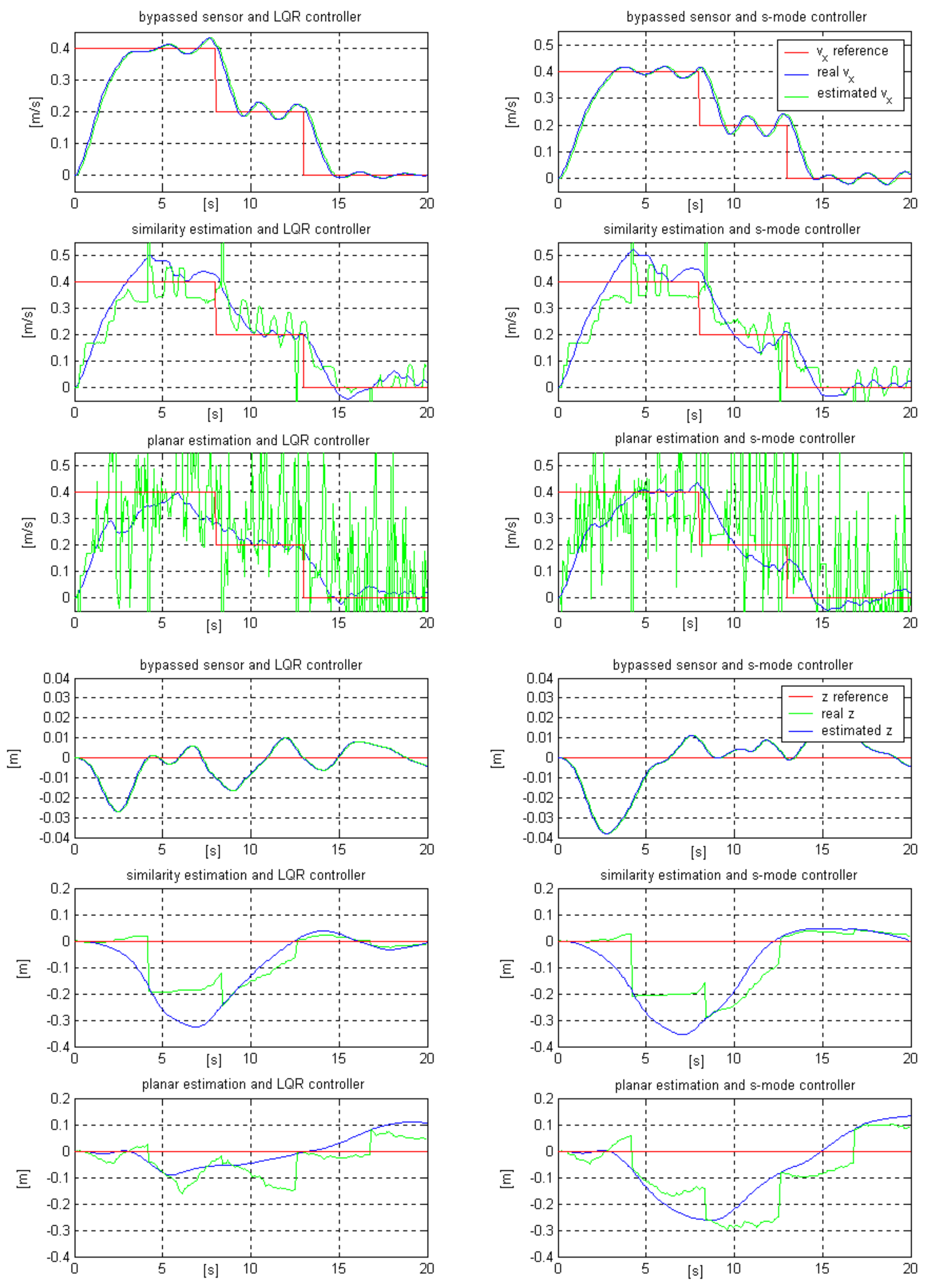


Figure C.2 - System response for the XZ controllers

The v_x response of the system with sensor bypass shows that the different controllers produce very similar system behaviours in ideal condition. As for the response when using the pose estimation algorithms, it is much better for the similarity homography model, as it proves to be much less noisy than full planar. Deviation from the desired values is a consequence of faulty sensor data and not the controller.

The initial building up on the error in z comes from the fact that the blimp tends to climb when accelerated forward. Moreover, because the initial velocity error is large this cripples the convergence of z, as they are actuated by the same applied force that is early driven to saturation. In spite of this, the largest altitude error observed is of 35 cm.

The oscillation observed, especially in the v_x velocity, is the result of the dynamics in the servo, which were neglected. When the controller tries to correct the altitude of the vehicle and displaces the servo from negative to positive angle or vice-versa, it introduces an actuation delay. Furthermore, as this displacement takes place, the propellers continue rotating inducing an undesirable force in the x direction. Therefore, this oscillatory behaviour is more common when the z error approaches zero. This problem can be attenuated by taking advantage of the fact that the thrusters have a natural dead-zone. If the desired actuation is sufficiently low (and it is for the case of these oscillations), the servo is able to achieve the desired position before the error has grown enough for the thruster to leave the dead-zone.

Bibliography

[1] Gracias, Nuno Ricardo, *Trajectory* Reconstruction *with Uncertainty Estimation Using Mosaic Registration,* Tese de Doutoramento, IST, Junho de 2003;

[2] Zwaan, Sjoerd van der, *Vision Based Station Keeping And Docking For Floating Robots,* Tese de Mestrado, IST, Maio de 2001;

[3] Gracias, Nuno Ricardo, Santos-Victor, José A., *Mosaic-based Visual Navigation for Autonomous Underwater Vehicles,* VisLab-TR 04/2001, - Robotics and Autonomous Systems (Elsevier), 35(3-4), Junho 2001;

[4] S. Hutchinson, G. Hager and P. Corke, A Tutorial on Visual Servo Control, *IEEE Transactions on Robotics and Automation* 12(5):651-670, October 1996;

[5] Fossen, Thor I., *Guidance and Control Of Ocean Vehicles*, John Wiley & Sons, 1994;

[6] Latombe, Jan-Cclaud, *Robot Motion Planning*, Klumer Academic Publisher, 2nd Printing, 1991;

[7] Cox, I. J. And Wilfang, G. T, *Autonomous Robot Vehicles*, Springer-Verlag, 1990;

[8] Beer, Ferdinand P. and Johnston, E. Russel, *Mecânica Vectorial Para Engenheiros - Dinâmica*, McGraw-Hill, 1998;

[9] J. Sequeira, *Robótica (Transparências das Aulas Teóricas)*, transparências da cadeira de Robótica leccionada no IST em 2001/2002, Lisboa, 2002;

[10] John J. Craig, *Introduction to Robotics, Mechanics and Control*, 2nd edition, Addison-Wesley Publishing Company, 1989;

[11] Heikkilä, J. & Silvén, O, *A Four Step Camera Calibration Procedure With Implicit Image Correction*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97), San Juan, Puerto Rico, p 1106-1112, 1997;

[12] O. Faugeras, *Three Dimensional Computer Vision – A Geometrical Viewpoint, MIT Press, 1993;*

[13] Nise, Norman S., *Control Systems Engineering*, third edition, John Wiley & Sons, 1994;

[14] Lemos, J. Miranda, *Notas da Disciplina de Controlo II*, transparências da cadeira de Controlo II leccionada no IST em 2001/2002, Lisboa, 2000;

[15] Slotine, Jean-Jacques E. and Li, Weiping, *Applied Non-linear Control*, Prentice Hall Inc., New Jersey 1991;

[16] Santos-Victor, José A., *Projecto de Sistemas de Processamento e Controlo Digital*, transparências da cadeira de Projecto de Sistemas de Processamento e Controlo Digital leccionada no IST em 2002/2003, Lisboa, 2002.