University of Padua

Master Thesis in Control System Engineering

Complex perception models applied to non-linear MPC based Motion Cueing Algorithms

Author:
Riccardo Gallina

Supervisor:
Prof.Dott.Ing. Alessandro Beghi
Co-Supervisors:
Dott.Ing. Mattia Bruschetta
Dott.Ing. Fabio Maran

A thesis submitted in fulfilment of the requirements
for the Master Degree

at the

Department of Information Engineering

October 2014

“If you never try, you’ll never know what you are capable of.”

John Barrow

“They always say time changes things, but you actually have to change them yourself.”

Andy Warhol

UNIVERSITY OF PADUA

Abstract

Engineering School
Department of Information Engineering

Master Degree

Complex perception models applied to non-linear MPC based Motion Cueing Algorithms

by Riccardo Gallina

Almost everyone had suffered from sickness on occasion when travelling as a passenger in an auto, ship, or aircraft. Motion sickness has a significant incidence in military and space operations, and is common in otologic disease. More than a century ago, Irwin (1881) noted that vestibular and visual sensory systems can play an important role in producing this disorder. However, despite the ubiquity of motion sickness in modern society, and extensive research efforts, the physiology underlying how the syndrome may function in the Central Nervous System (CNS), has not yet been particularly well defined. As it consequence, the cause of motion sickness is still explained primarily in psychophysical terms.

The objective of the present effort is to analyse some mathematical models for conflict generation in motion sickness and to implement them in an efficient way that can allow their use in nowadays practical applications, such as driving simulators. The resulting model described here in detail, has been presented earlier in preliminary form (Merfeld, 1993; Newman, 2009). The development of a mathematical representation of the conflict theory for motion sickness may lay some useful qualitative considerations to the motion sickness problem.

Acknowledgements

Many people have contributed in recent years to support me in college.

I thank Professor Alessandro Beghi, for the availability and for the thesis opportunity he had offered me. I thank also Fabio Maran, for his time and for the accuracy shown in the required interventions, and, in particular, Mattia Bruschetta without whom contribution there would not be such results in this thesis.

Then, I thank all engineering colleagues with whom I shared this college experience, both in department and during train journeys.

Last but not least, I thank all my family, that has always supported and encouraged me, and without which I would not obtain such results all over these five years.

Sincere thanks to you and more in general to all those that have always believed in me!!

Contents

List of Figures

List of Tables

Abbreviations

OKN	**O**pto**K**inetic **N**ystagmus
OKAN	**O**pto**K**inetic **A**fter **N**ystagmus
OVAR	**O**ff **V**ertical **A**xis **R**otation
GIF	**G**ravito **I**nertial **F**orce
CNS	**C**entral **N**ervous **S**ystem
SCC	**S**emi **C**ircular **C**hannel
OTO	**OTO**lith
MCS	**M**otion **C**ueing **S**ystem
MCA	**M**otion **C**ueing **A**lgorithm
MPC	**M**odel **P**redictive **C**control
NMPC	**N**on-linear **M**odel **P**redictive **C**control
SISO	**S**ingle **I**nput **S**ingle **O**utput
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
FIR	**F**inite **I**mpulse **R**esponse
MHE	**M**oving **H**orizon **E**stimation
QP	**Q**uadratic **P**rogram
DOF	**D**egree **O**f **F**reedom
OCP	**O**ptimal **C**ontrol **P**roblem
ACADO	**A**utomatic **C**ontrol **A**nd **D**ynamic **O**ptimization
QPP	**Q**uadratic **P**rogramming **P**roblem
KF	**K**alman **F**ilter

Symbols

a	linear acceleration	ms^{-2}
v	linear velocity	ms^{-1}
p	linear position	m
g	gravity vector	$9.81ms^{-2}$
g_v	visual gravity vector	$9.81ms^{-2}$
f	gravito-inertial force	$9.81ms^{-2}$
$\hat{a}$	estimated linear acceleration	ms^{-2}
$\hat{v}$	estimated linear velocity	ms^{-1}
$\hat{p}$	estimated linear position	m
$\hat{g}$	estimated gravity vector	$9.81ms^{-2}$
$\hat{f}$	estimated gravito-inertial force	$9.81ms^{-2}$
e_f	gravito-inertial force direction error	$9.81ms^{-2}$
e_ω	angular velocity error	$degs^{-1}$
e_{gv}	visual gravity error	$9.81ms^{-2}$
$e_{\omega v}$	visual angular velocity error	$degs^{-1}$
N_p	prediction horizon	
N_c	control horizon	
ω	angular velocity	$degs^{-1}$
ω_g	gravity weighting angular velocity	$degs^{-1}$
θ	angular position	deg
α_f	perceived gravito-inertial force	$9.81ms^{-2}$
α_ω	perceived angular velocity	$degs^{-1}$
$\hat{\omega}$	estimated angular velocity	$degs^{-1}$
$\hat{\theta}$	estimated angular position	deg

$\hat{\alpha}_f$	estimated perceived gravito-inertial force	$9.81ms^{-2}$
$\hat{\alpha}_\omega$	estimated perceived angular velocity	$degs^{-1}$

Chapter 1

Introduction

Daily human activity includes complex orientation, postural control, and movement coordination, as explained also by Selva [1]. All these tasks depend upon the human perception of motion. The non-auditory section of the inner ear, the *vestibular system*, is recognized as the prime motion sensing center. It represents an inertial measuring device which allows us to sense, in the absence of external sensory cues (vision, etc) self-motion with respect to the six degrees of freedom in space (three rotational and three translational).

The information from the vestibular apparatus is used in three ways:

- to provide a subjective sensation of movement in three-dimensional space
- to maintain upright body posture (balance)
- to control the muscles that move the eyes direction, so that in spite of the changes in head position which occur during normal activities such as walking or running, the eyes remain stabilized on a point in space.

Several scenarios illustrate these points. For instance, if a cat is dropped upside down, it will land right side up on all four paws. If a newborn infant is tilted backward, its eyes will roll downward so that its gaze remains fixed on the same point. If you, as you read this introduction, shake your head rapidly from side to side, the print nonetheless will stand still. Each of these scenarios is an example of how a healthy balance (vestibular) system compensates for daily changes in our spatial orientation.

The vestibular system is comprised of two primary sense organs:

- the semicircular canals (SCCs), which detect angular accelerations of the head

- the otolith organs (OTOs), which respond to linear accelerations of the head and to gravity.

Thus, vestibular sensors provide information to the brain regarding our body's position and acceleration in space, with sensing capabilities that are compatible with everyday movements relative to the surroundings, and hence play central role in spatial orientation.

Spatial orientation can be defined as one's perception of body position with respect to a reference frame. This process involves two main sensory modalities, the *vestibular system* and *vision*, but proprioceptive and auditory inputs also come into play. The control of spatial orientation during navigational and locomotion tasks requires a dynamic updating of the representation of the relations between the body and the environment, i.e. spatial orientation normally entails both the subconscious integration of multisensory cues and the conscious interpretation of external information. Therefore, the Central Nervous System (CNS) uses information coming from multiple sensors to come up with a representation of how the body is moving and is oriented in space.

The results of this "spatial orientation" process are usually satisfactory in most of everyday life situations. However, when technology achievements began to expose humans to new and artificial situations such as sustained accelerations in fighter airplanes, microgravity environment in spacecrafts or *motion cueing* in driving simulators, our ability to correctly estimate our position and motion became limited. As a matter of fact, as the number of fighter airplane accidents due to technical failure keeps decreasing, human errors have been proven to be a limiting factor to safety. That is, the advent of aeronautics flight has not only involved a new demand on human organism but also the ability for pilots to deal with a high workload environment and a complex instrument panel. Furthermore, in some circumstances, e.g. when flying in clouds or at night, pilots may not have the possibility of seeing external references. As a result, they are constantly liable to introduce conflict between their internal feeling of orientation and the real one, and hence to experience a case of spatial disorientation which is a phenomenon attributed to 15 to 30% of all aircraft fatalities in flight [2]. All these considerations have lead number of researchers to model human spatial orientation.

Mathematical models for three-dimensional human spatial orientation have continued to evolve over the past four decades. Several models exist and have been developed using multiple computational approaches such as linear systems analysis, the concept of internal models, observer theory, Bayesian theory, Kalman filtering and particle filtering. A review of these approaches has recently been written by MacNeilage [3]. Different features can be distinguished among these models: some of them are restricted to one-dimensional space, others take into account motions in three-dimensional space; some incorporate visual cues, while others only model vestibular response in the dark; and some work for large head tilts whereas others do not.

1.1 The purpose

The aim of this work is to test a different and more sophisticated approach to the motion cueing problem, with particular interest on driving simulators applications. Most of the work focuses on obtaining as final model a non-linear one with complex dynamics, that could allow to get good results in the motion perception and spatial orientation of the driver during a driving session.

Another important aspect of this thesis is the computational cost of the tests: it has to managed in a smart way. The idea is to work in real-time, so it is not used only Matlab code, but a more efficient toolkit, the `ACADO toolkit`, that allows for better computational performances converting the code in faster programming languages. So first of all it is necessary to take confidence with this new tool.

In the end, a comparison with previous works is shown. In particular, it is taken into consideration the implementation of a motion cueing algorithm through a linear Model Predictive Control (MPC) that uses some linear filters to simulate the platform and the vestibular system. In this way, it is possible to sum up some qualitative considerations about the complex non-linear model presented in this thesis work.

Chapter 2

Motion Cueing Algorithms

Linear and rotational accelerations, which exert forces on the driver's body, are called *motion cues*. In the automotive field, such as in dynamic driving simulators, these should convince the driver that he is driving a real car. The algorithms, which are responsible for generating these cues, are named *Motion Cueing Algorithms*, (MCAs) [4].

Strategies are required to take into account restrictions on platform movements and, at the same time, to consider the human perception of motion. The limited workspace, for example, makes it impossible to generate long lasting acceleration cues. To avoid this problem, most algorithms are based on three phases.

The first one is the discrimination of the frequency content of the input acceleration signal in a high frequency part and in a low frequency part. In the second phase, the high frequency signal is filtered by the washout action (a cascade of filters, see Section 2.1), which adds a drift to the executed motion, so that the platform is always pulled back to the centre of the work space. In the last phase, the low frequency signal is transformed by the tilt-coordination (see Section 4.2) into an angle that slowly tilts the platform and the visual scene, so that gravitational vector differs from the body z-axis. The part of the vector that is aligned with the platform axis in the longitudinal (x) and lateral (y) direction creates the illusion of sustained accelerations.

However, these methods to obtain a good motion cueing are subject to false cues such as wrong platform movements, which disagree with the experience of real car driving. These contradictory motion cues could cause *simulator sickness*. If the simulated platform movement does not in agree with the visual car movement, then a sensory conflict emerges. The body responds, among others, with dizziness and disorientation. Therefore, it is important to set up the algorithms accurately, in order to avoid these harmful effects.

2.1 Different approaches to Motion Cueing

To make the simulator move in an intelligent way it is used a Motion Cueing System (MCS), which aims to transform the input acceleration references $r(t)$, to possible signals to be provided to the actuators of the simulator platform. The algorithms of a MCS have therefore constrained objectives that have to be met:

- it has to make the driver perceive a feeling of motion that is as close as possible to the reality
- it has also to keep the platform within its limits,

as explained by D'Ambrosio and Baseggio [5] [6].

2.1.1 Classical Washout Algorithm

The first algorithm of this type was proposed in the early 70s by Schmidt and Conrad who developed the classical strategy for Motion Cueing which was going to calculate the acceleration of the simulator along all its degrees of freedom. The classical MCS algorithm, made by frequency filters, is structured in the following way:

1. low frequencies have to be removed from longitudinal accelerations by an high-pass filter, and then the result is integrated twice in order to obtain the position control of the platform as an output of the algorithm;
2. low-frequency accelerations are extracted from the reference by a low-pass filter and from these, the angle for tilt coordination is calculated, that will be added to the output control for the angular positions;
3. the platform is ensured to its neutral position by filtering through an high-pass filter the resulting control position.

The latter filtering, often called Motion Washout [7], is necessary to prevent the saturation of the actuators which then could lead to make the driver perceive false dynamics of the vehicle. Another type of filtering, called anti-backlash filter, can be added to reduce some non-optimal responses following the high-pass filtering (Reymond, 2000) [8].

Using this type of algorithm and filtering out the low frequencies of the reference, cause the resulting acceleration to be extremely limited. This is caused by the setting of the filter parameters (such as gain and cut-off frequency) that will be set from session

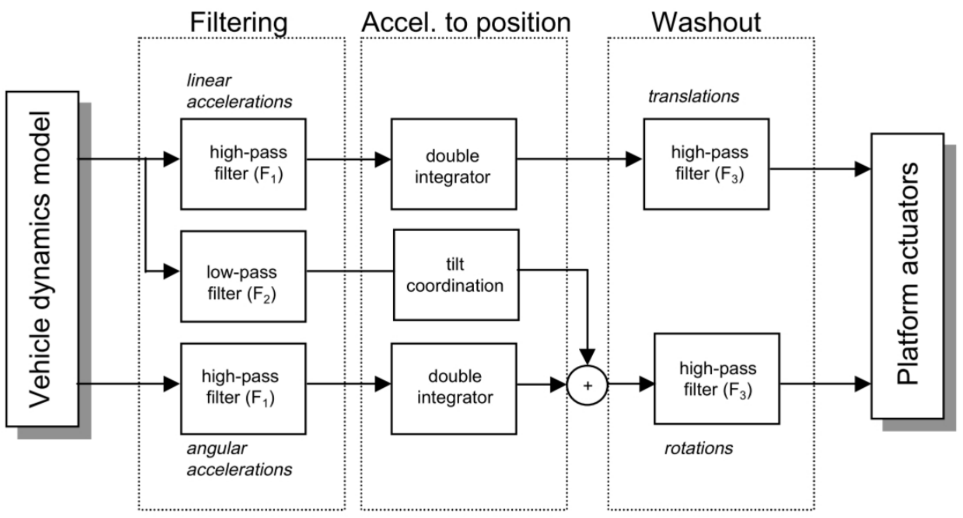


FIGURE 2.1: Classical washout algorithm.

to session, depending on the circuit and driving style, taking into consideration that the cueing must ensure that the actuators will never saturate. When, for example, it happens that the cockpit is near the limit of the platform position, the algorithm has two solutions: either to stand still with the cab playing only with the angular accelerations, waiting for an in-phase acceleration in order to come back to its neutral position, or go in opposite-phase and make the pilot perceive an acceleration in contrast to the one received by the visual system, thus causing him a feeling of sickness.

In both cases, the solutions do not meet the task of the motion cueing algorithm and then it is necessary to set up the filter taking into account the worst case, thus to make the maximum acceleration input remain within the thresholds of the platform's actuators. As a logical consequence, during a driving session where there are only "small" accelerations, it will be used only a small portion of the possible motion workspace of the simulator.

2.1.2 Adaptive washout algorithm

To overcome the problem of not exploiting all the space available to the simulator, Parrish and Dieudonne [9] proposed an adaptive method, further developed in the next years. Based on the classical MCS, instead of using only the strategy in the frequency domain, they also considered the time domain: the filtering process remains the same as in the classical approach, but in this case the parameters are not constant but are set up again at each step.

At each sampling instant, the gain and the cut-off frequency of the classic MCS approach are derived from the minimization of a quadratic cost function:

$$V_k = (r_k - a_k)^2 + \omega_1 \cdot v_k^2 + \omega_2 \cdot p_k^2$$

The first term of V_k is the squared error between the acceleration of the real vehicle r_k and the one of the platform a_k. The remaining two terms are related to the speed v_k and the position p_k of the platform.

Compared with the previous version, this adaptive strategy try to exploit in a more efficient way the work space during "regular" driving conditions, i.e. with accelerations smaller than the values on which the filters are set. The weights ω_1 and ω_2 define a trade-off between the developed feeling of motion and the limits imposed by the actuators: an increase of ω_1 and ω_2 will penalize displacements and high speeds while reducing them will go into advantage of the square error minimization between the real and the platform acceleration. In any case, the setting of these parameters must be done off-line whereas the specific driving of the pilot and the type of circuit on which the driving simulation is performed must be taken into consideration.

2.1.3 Optimal washout algorithm

In 1982 Sivan and Ish Shalom [10], developed an optimal algorithm which, combined with linear low-pass and high-pass filtering as in the classic MCS, calculates the controls to be given to the platform by minimizing a global cost function:

$$V[u(t)] = \int_0^{\infty} [(\tilde{r}(t) - \tilde{a}(t))^2 + \omega_1 \cdot v^2(t) + \omega_2 \cdot p^2(t)]dt$$

subject to

$$x(t) = Ax(t) + Bu(t)$$

with $x = [p\ v\ a\ \tilde{a}]^T$ and where (A,B) are the matrices of the model of motion perception, $\tilde{a}$ is the acceleration perceived by the driver inside the car and $u(t)$ is then control input of the system. To calculate the minimum of the cost function is necessary to know the reference along an entire driving session for the time interval $[0, \infty]$.

In this last strategy, it can be found, in addiction to the model of the considered system, the first use of a model of motion perception that was proposed in earlier years by Zacharias (1978); this was necessary to reproduce the motion feeling in a structure with well-known constraints, and so the purpose is to mislead the perception of the pilot in order to follow the trajectory but not only using longitudinal accelerations but also through the tilt-coordination.

Chapter 3

Model Predictive Control

In this chapter it is introduced the Model Predictive Control (MPC) and its fundamental characteristics are described. There are a lot of works about MPC, as [5], [11], [12], [13], [14], [6], but for a detailed analysis it is suggested to read Maciejowski and Wang's books [12] [11].

The predictive control is a technique born for industrial purposes at the end of 70s, further developed by researchers, that found a lot of useful fields in which it could be applied, thanks to its peculiarity to solve constrained control problems. This technique provides the optimal inputs to a system, that minimize a particular cost function. The fundamental idea is to compute the control (input) such that an open-loop and finite-horizon optimal control problem is solved with respect to the constraints, within a fixed time horizon. From the sequence of optimal inputs obtained, only the first one is applied and then the optimal problem is iterated using the new state of the system as the initial condition.

3.1 Pros & Cons

The great success of MPC is certainly due to the many advantages that its usage entails, especially in the industrial sector applied to very complex problems, e.g. the ability to handle constraints on the variables of the problem, namely the outputs, inputs and changes of the inputs of the system. This fact is of crucial importance in practice, because a lot of applications involve limited quantities. Constraints can also be divided into *hard* and *soft*: the first are bounds that absolutely do not have to be violated, while the others can be violated, despite losing in terms of cost. The *predictive* control is also an extremely versatile tool: it adapts to SISO as well as complex MIMO problems or

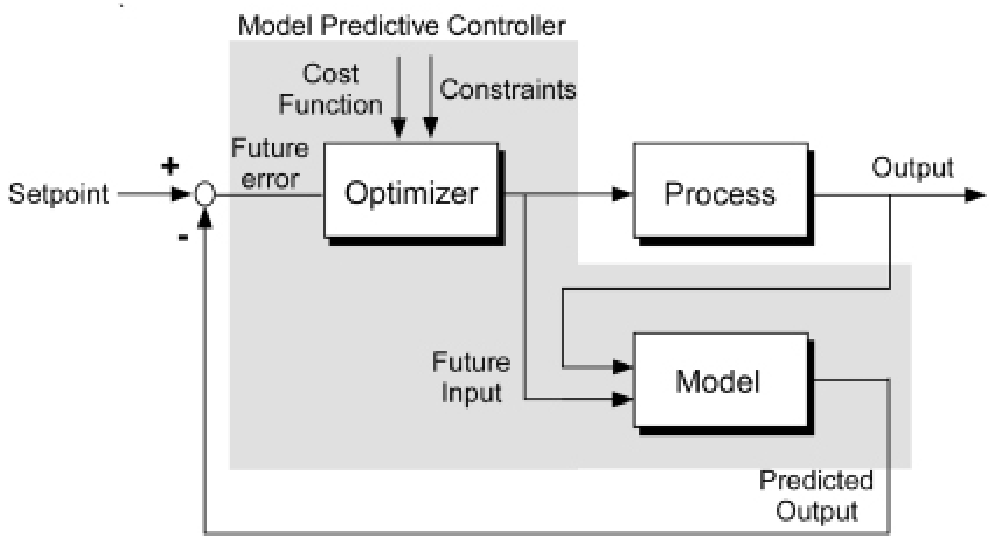


FIGURE 3.1: Model predictive control scheme.

non-minimum phase or even unstable. The MPC can be interpreted in different ways: by its nature it is an open loop control, but in some cases can be thought as a component of the feedback; it can also be used in a feedforward control to compensate for the input disturbances of the system.

The method to obtain the optimal input, as already mentioned, occurs through the choice of a functional cost, thus it is possible to choose this functional according to the needs of the problem and give emphasis on the variables that require more attention. The management of the constraints also allows, in some cases, to work close to the limits but in safety, thus exploiting further the potential of the plant.

The predictive control obviously has some disadvantages. Until a few years ago, the biggest difficulty was the computational cost. Since the MPC is an optimal control problem with constraints to be solved at each sampling step after updating the system, in the absence of appropriate calculation tools and dealing with computationally expensive problems, it is easy to understand how the MPC had some intrinsic limitations in its nature. However, with the improvements of the technology and with more and more computing power, this defect is fading very quickly.

Another problematic aspect is the *stability*: it is difficult to ascertain in a problem with constraints and therefore can not be rigorously formalized, but as long as it can, the controller exerts its function and therefore behaves in a stable manner, even if a sharp variation of the value of the involved variables could lead to undesired behaviours. A final critical aspect is the fact that the MPC uses an internal model of the plant that has to be controlled, on which it has to make the prediction. Obviously, this model may not be accurate or may change, so the evolution predicted by the controller may not be consistent with the one actually followed by the plant.

3.2 The algorithm

In this section it is described, in a qualitative manner, the necessary steps to set up a predictive control.

3.2.1 Model and prediction

First a model of the system is needed. Initially, the model predictive control was based on FIR or step response models, which, however, are able to describe only stable implants, and are often of a high order. An improvement is the use of transfer functions, which, therefore, can also be applied to unstable systems, but they are difficult to manage in the case of multi-variable problems. These defects have led over the years to affirm the use of *state space models*, which allow to take advantage of linear systems theory and efficiently manage multi-variable systems. They also permit to easily model errors and noises and moreover to take advantage of the theory of statistical filtering for systems where the state is not accessible. Once chosen the model of the system under analysis, the prediction H_p and control H_c are fixed and the evolution of the system in this time window is calculated, writing the future outputs $y(t+k|t),\ k = 1, \dots, H_p$ as a function of the future inputs $u(t+k|t),\ k = 0, \dots, H_c - 1$.

3.2.2 Cost function and control computation

The optimal input is calculated minimizing the *cost function*, or *objective function*, chosen on the basis of contextual needs of the problem. The basic objective is to follow a reference trajectory, while respecting of the constraints. A possible cost function is the following:

$$J = \sum_{j=1}^{H_p} \delta(j)\, [\tilde{y}(t+j|t) - r(t+j)]^2 + \sum_{j=1}^{H_c} \lambda(j)\, [\Delta u(t+j-1)]^2$$

It minimizes the error $\tilde{y}(t+j|t) - r(t+j)$ and the cost in terms of change of the input $\Delta u(t+j-1)$, depending on the values of the weights $\delta(j)$ and $\lambda(j)$. It is also emphasized that the choice of the functional is not unique but can be arbitrary, the important thing is that it must be quadratic, as explained in Section 7.4.1. Because of the constraints, the minimization translates itself into a quadratic programming problem: in the case of absence of constraints the solution is obtained in an analytical way. Often the solution to the quadratic programming problem is not so immediate because the complexity of the problem is quite high; so, it is usually imposed $N_c < N_p$ and assumed that the input signal does not undergo more changes after N_c steps: $\Delta u(t+j-1) = 0,\ j \in [N_c, N_p]$. In this way the size of the problem is reduced with immediate impact on the computational complexity.

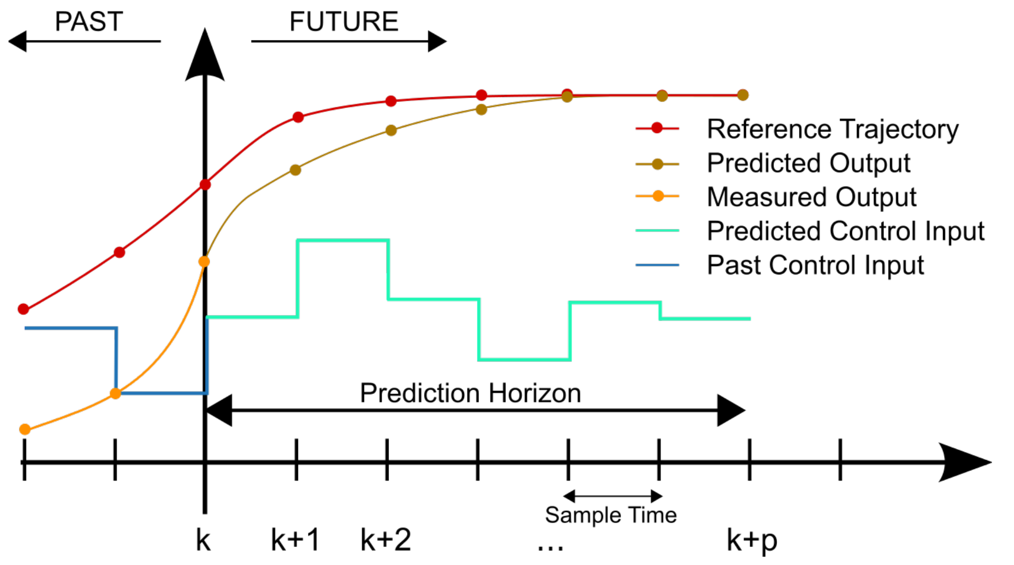


FIGURE 3.2: An example about how a model predictive controller works.

3.2.3 Control application and *receding horizon* technique

Since the model of the plant which is used by the MPC may not be accurate or there may be not measurable interferences, the whole sequence of the calculated control signal is not applied, but only the first element, $u(t|t)$, is given to the plant while the the rest is discarded. At the following iteration of the algorithm there will be the new values of the outputs and the system states and these variables will be used to repeat the whole procedure. In the end this process produces a new sequence of inputs of which it will be applied only the first element, $u(t+1|t+1)$, which in general will be different from the one that was calculated and discarded in the previous step, $u(t+1|t)$. This technique is called *receding horizon*, since the prediction horizon is always of the same length but it is moved forward by one step at each iteration. The fact that the new output is available to perform the optimization, implies that the model of the system under analysis is strictly proper, or that the output $y(k)$ depends only on the past inputs and not on the current $u(k)$.

Chapter 4

Human Perception System

The goal of a simulator is to provide the perception of what happens in a real driving situation, even if the platform moves in a different way in comparison to how the car would do on the road or on the track. Then it is necessary to model the human perception system to understand the relationship between real and perceived linear accelerations and angular velocities, [14]. In this way, it is possible to generate input signals that trick the organism, making the pilot feel in the platform the same feelings that he would perceive in a real situation. The sensors that perceive human motion belong to the vestibular system, which acts in a coordinated way with the vision. In this chapter it is described, therefore, the vestibular system and the key strategy, to make the perceptions as much real as possible, is presented: the *tilt coordination*.

4.1 The vestibular system

As explained by Selva [1], the inner ear is divided into two parts: the cochlea serving auditory function, and the vestibular system which contains the sensors providing information of body orientation and balance in three-dimensional space. The motion of the body is thus detected by the vestibular system, encoded as an electrical signal, and transmitted to the brain through the vestibular nerve. The brain then integrates vestibular, visual, and somatosensory inputs to estimate the orientation and motion, and consequently elicits eye, head, or body movements that will stabilize gaze and maintain balance.

There is one vestibular system on each side of the head, in close approximation to the cochlea. Due to its specific structure, this system is also called the labyrinth, see Figure 4.1. One distinguishes between the bony labyrinth and the membranous labyrinth.

The bony labyrinth is a complex cavity tunnelled in the temporal bone of the skull. Its structure forms three ducts, the semicircular canals, that converge toward a larger central part called “the vestibule”. The membranous labyrinth is enclosed in this osseous labyrinth, and is suspended in a fluid called “the perilymph” [15]. In birds and mammals, fine connective tissue filaments suspend the membranous duct within the osseous canal. The filaments serve to anchor the membranous labyrinth to the temporal bone such that the gravito-inertial acceleration experienced by the sensory organs could be expected to be nearly identical to that experienced by the temporal bone. To date, there are no experimental data to suggest significant relative motion between the temporal bone and the membranous labyrinth (Rabbitt, [16]). The membranous labyrinth is also filled with fluid known as ”the endolymph”, physically a water-like liquid. Each side of this bilateral system consists of two types of sensors: a set of three semicircular canals sensing rotation movement, and two otolith organs (the saccule and utricle) which sense linear movement and head tilt.

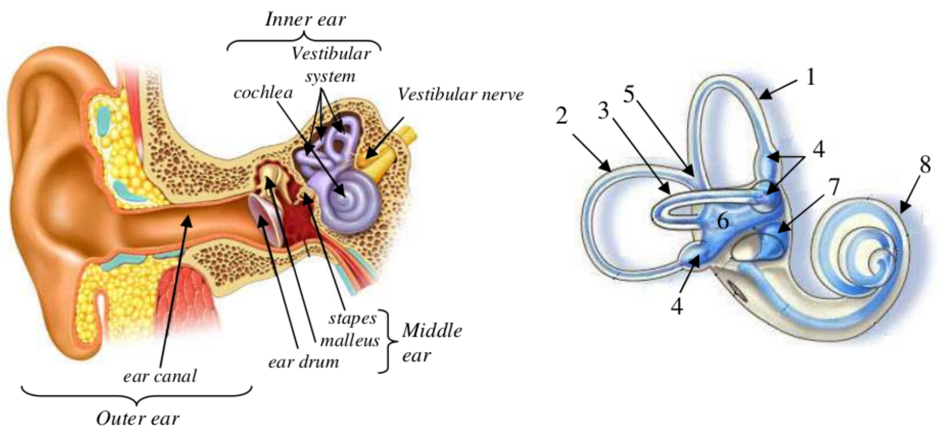


Figure 4.1: Visualization of the inner ear. 1) Anterior canal, 2) posterior canal, 3) lateral canal, 4) ampulla of each canal, 5) common crux, 6) utricle, 7) saccule, 8) cochlea.

4.1.1 The semicircular canals

The semicircular canals are commonly referred to as the lateral canal, also called horizontal canal, and the posterior and anterior canals, which constitutes the vertical canals. These latter have a common duct called the *common crux* for about 15% of their length. The canals are oriented in almost mutually orthogonal planes. The lateral canal lies in a plane elevated about 30 degrees from the horizontal plane, while the two others are arranged in diagonal planes which subtend roughly 45 degrees relative to the frontal and saggital planes of the skull, see Figure 4.2. Thus, the anterior canal on one side of the head is parallel to the posterior canal on the other and vice versa, whereas the horizontal canals of both inner ears lie in the same plane. Because most head movements are not in a single SemiCircular Canal (SCC) plane, and also because of the imperfect orthogonality of the three canals, the labyrinth usually resolves a given head rotation into three components. That is, endolymph motion in each canal measures component of the head's rotational velocity in the plane of that canal. It has also been shown that each canal admits a specific direction of stimulation, which maximizes the excitation: the lateral, anterior and posterior canals primarily sense yaw, roll and pitch respectively (Rabbitt, [17]).

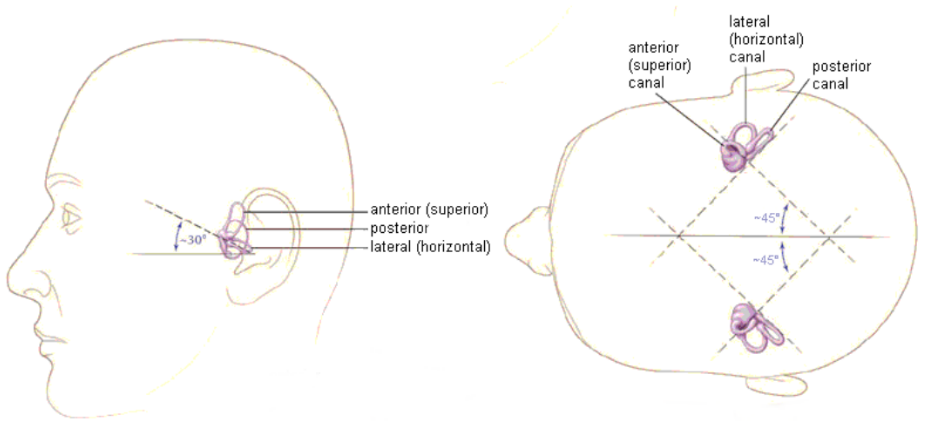


FIGURE 4.2: Orientation of the semicircular canals.

The set of canals constitute a very small fluid-filled system the size of a pea. They approximately form a circular path of 3.2mm radius and have a cross section radius along their slender part of about 0.16mm (Curthoys et al., [18]). The study of Curthoys and Oman probably constitutes the most thorough investigation concerning the dimensions of the human semicircular canals. From micro dissected specimens, they were able to provide measurements of the sizes, cross-sectional shapes and areas all around the path of fluid flow through the horizontal semicircular duct, ampulla and utricle.

At one location in each canal, and more precisely near the utricle, the canal cavity swells to form a bulbous expansion known as the ampulla that contains a transverse ridge of sensory epithelium, the crista. The epithelial surface of the crista contains thousands of sensory hair cells and surrounding supporting cells. Hair cells and supporting cells are found not only atop the ridge (crest) of the crista, but also down its sloping flanks. Hair cell sensory cilia project a short distance into tiny channels in the cupula, a gelatinous structure that extend upward from the surface of the crista all the way to the vault (roof) of the ampulla. The channels in the cupula material may be created as cupula material is secreted upwards from the supporting cells surrounding the each hair cell. The cupula effectively forms a thick diaphragm that completely occludes the canal lumen above the crista, and covers the entire sensory surface on the crest and both flanks. As detailed later, it is now believed that the cupula appears attached to the ampulla around its entire periphery.

When the head is subjected to an angular acceleration, endolymph inertia creates a hydrostatic pressure that deforms the cupula. Bending of hair cell stereocilia then initiates a complex transduction process in hair cells and vestibular afferent neurons. The nervous signal is finally transmitted to the brain and a sensation of motion results. At a constant rotation rate, the endolymph in the canals tends to catch up with the rotation of the head due to the viscosity, eliminating the relative movement. Eventually, as long as the rotation rate remains constant, the cupula returns to a vertical position due to its elastic properties and the sensation of motion eventually ceases.

4.1.2 The otolith organs

The otolith organs, the saccule and utricle, are situated between the semicircular canals and the cochlea, and are approximately perpendicular to each other, see Figure 4.3 a. They are the elements of the vestibular system that provide linear motion sensation in human and mammals. They are sensitive to the direction of the Gravito-Inertial Force (GIF) applied to the head, and consequently respond to both linear acceleration and tilting of the head with respect to gravity. The saccule is dedicated to measuring primarily the vertical component of the GIF with respect to the head, whereas the utricle measures primarily the horizontal component. As stated by Einstein's equivalent principle, all linear accelerometers must measure both linear acceleration and gravity (Einstein 1908). Therefore, the otolith organs cannot discriminate between acceleration and tilt, requiring additional sensory information to resolve this ambiguity.

Both the saccule and utricle are flat layered structures. The top layer, which is in contact with the endolymph, consists of calcium carbonate crystals called otoconia, the middle layer consists of a gelatinous matrix called the otholitic membrane, and the bottom layer consists of a bed of hair cells known as the macula that is rigidly attached to the skull and therefore moves with the head. The hair cells are anchored in the macula whereas their cilias extremities are embedded in the otolithic membrane.

The orientation of the hair cell bundles is organized relative to a region called the striola, which demarcates the overlying layer of otoconia. The striola forms an axis of symmetry such that hair cells on opposite sides of the striola have opposing morphological polarization. Thus, a tilt along the axis of the striola will excite the hair cells on one side while inhibiting the cells on the other side.

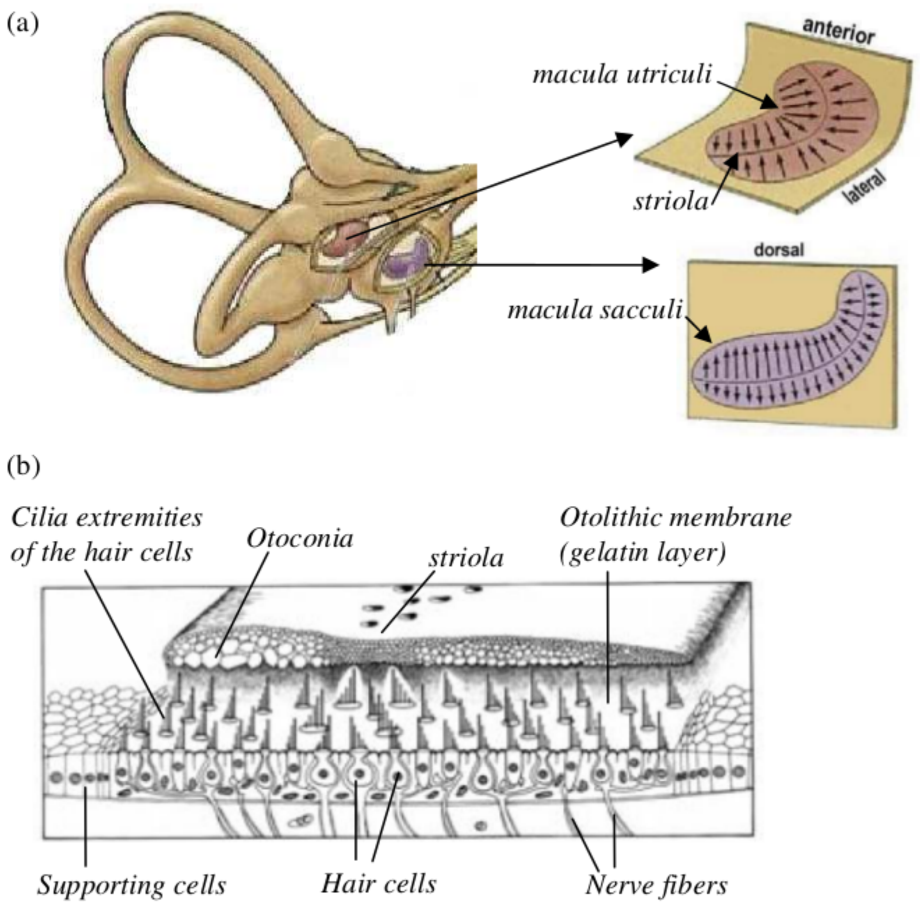


FIGURE 4.3: Physiology of the utricular macula. (a) Location of the utricle and saccule and orientation of the hair cells on the maculae of the otolith organs. (b) 3D perspectives of a macula.

4.2 Tilt coordination

As Daniele D'Ambrosio has experienced in his thesis work [5], the simple translation of the platform along the x and y axes is not able to reproduce the accelerations during a normal driving simulation. However, thank to the fact that the human body is unable to distinguish from translational and gravitational accelerations with the only contribute of the maculae, it is possible to deceive the human perception system by tilting the platform by an accurate angle. In this way, the simulator gives the driver a feeling of acceleration that has to be added to the one provided by the longitudinal motion. This technique is called *tilt-coordination.*

As shown in Figure 4.4, tilting the cockpit by a pitch angle makes the longitudinal and gravitational forces be expressed by their respective components along the x and z axes. In this way, the resulting perceived force is the sum of the their longitudinal components (x axis). More in detail, giving an inclination of a pitch angle θ and a roll angle ϕ (the

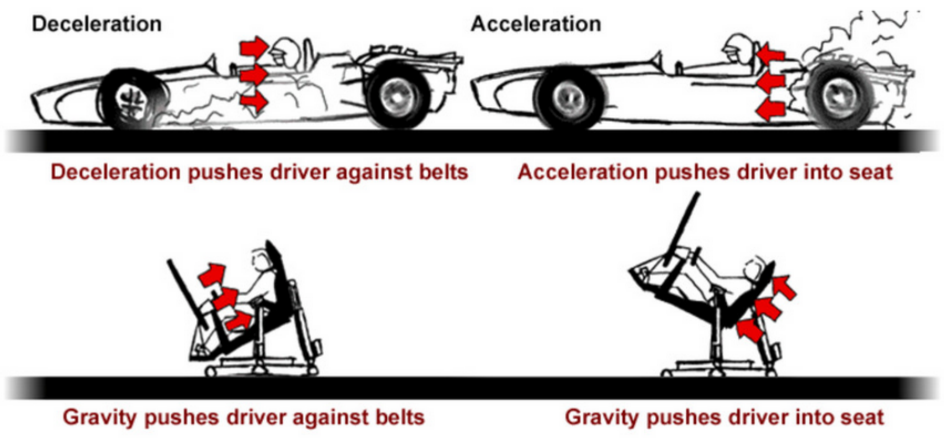

Deceleration
Acceleration
Deceleration pushes driver against belts
Acceleration pushes driver into seat
Gravity pushes driver against belts
Gravity pushes driver into seat

FIGURE 4.4: Tilt coordination explanation.

angle of yaw is indifferent to the tilt coordination), the gravitational acceleration vector g, expressed in the platform frame through appropriate rotation matrices, is expressed as:

$$g_s = R_x(\phi)R_y(\theta)g_i = \begin{bmatrix} -g\ sin\theta \\ g\ cos\theta\ sin\phi \\ g\ cos\theta\ cos\phi \end{bmatrix}$$

where the subscripts s and i are used to indicate vectors related to the platform and to the inertial frame respectively. Then, it is defined the specific force $f_s = a_s - g_s$, that is the acceleration desired unless the gravity force, which is unrelated to the platform and therefore it is not necessary to replicate. This specific force can be expressed by its components as:

$$f_s = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} a_x + g\ sin\theta \\ a_y - g\ cos\theta\ sin\phi \\ a_z - g\ cos\theta\ cos\phi \end{bmatrix} \approx \begin{bmatrix} a_x + g\theta \\ a_y - g\phi \\ a_z - g \end{bmatrix}$$

where the last approximation has been made under the assumption of small angles. The technique of the tilt coordination, as well as making possible to obtain perceived accelerations greater than those simulated only by the longitudinal shift, allows to simplify the problem: the controller chases the high frequencies of the reference by longitudinally shifting the platform and the low frequencies by tilting it. This solution to the problem of motion cueing is pretty simple and basic. It does not fully exploit the potential of the platform, however it is a good starting point and it is the technique upon which most of the motion cueing algorithms on the market work.

Chapter 5

Sensory Conflict Model

“Velocity storage” is a concept which is used as a central component in almost all classical control models of spatial orientation. This element was developed when it was observed that per and post rotatory nystagmus[1] last well beyond the activity of the first order afferents from the semicircular canals. At nearly the same time, OptoKinetic After-Nystagmus (OKAN) was noted as another response which lasted beyond the visual stimulation. Furthermore, it seemed that the time course of these responses was similar. The velocity storage hypothesis proposed that a single neural element was responsible for the extension of vestibular nystagmus and for OKAN.

The approach which Merfeld presents in [20], [21], is analysed in the following pages. Specifically, he developed the internal model approach suggested by Young [22], using the observer theory approach used by Oman [23]. The developing of the model starts using a very general non-mathematical description; then, a series of simplifying assumptions reduce the model to a one-dimensional linear model. This simple version of the sensory conflict model is investigated using two examples. It is then generalized until the model includes again three dimensions and some inherent non-linearities.

Figure 5.1 shows a very general block diagram which presents the philosophy underlying the development of the entire model. “Desired orientation”, the primary system input, is compared to the “estimate of orientation” to yield an orientation error. A “control strategy” is applied to the orientation error to yield a “motor command”. The motor command is relayed to the muscles which are represented as part of “body dynamics” to yield the “actual orientation”. The actual orientation is measured by the sensory organs that produce the physiological output, the “sensory afference”. A copy of the efferent signal (“efferent copy”) is sent to an internal model of body dynamics (including muscle

[1]condition of involuntary eye movement, acquired in infancy or later in life, that may result in reduced or limited vision. Due to the involuntary movement of the eye, it is often called ”dancing eyes”. [19]

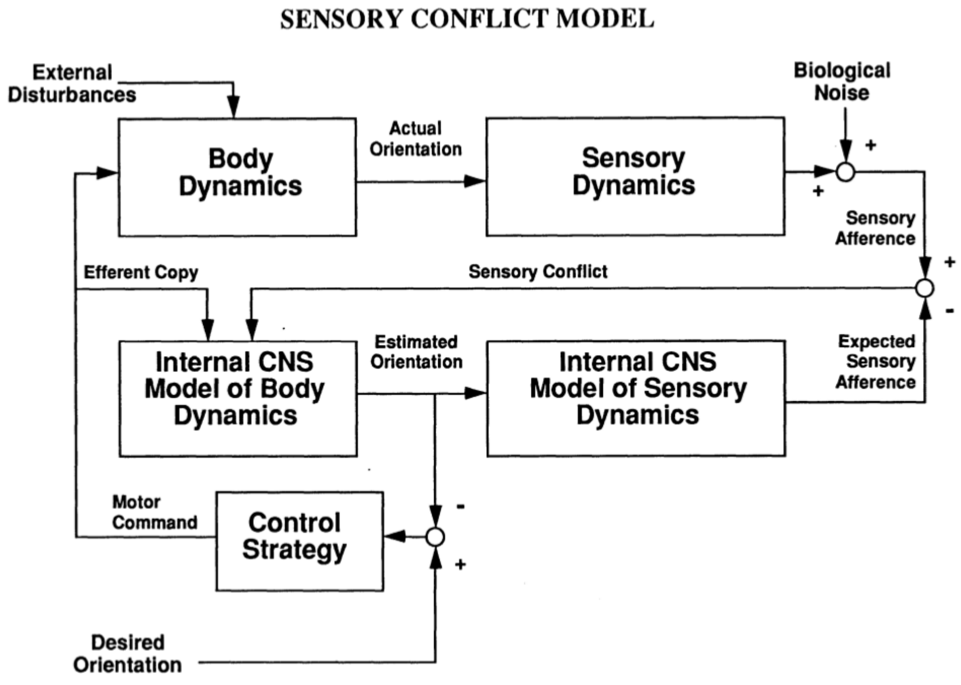


FIGURE 5.1: Sensory conflict model by Merfeld.

dynamics) to yield the "estimated orientation". This estimate of orientation is sent to a model of sensory dynamics to give an "expected sensory afference". A difference between the sensory afference and the expected sensory afference indicates "sensory conflict". The sensory conflict returns to the internal model of body dynamics to drive the estimated orientation toward the true orientation.

5.1 One-dimensional model

In Figure 5.2 the model is simplified by developing a one-dimensional (z axis) linear observer. Moreover, the model takes into account only passive motion, so there is no control strategy, such as the subject can only undergo external disturbances and forces. The matrix transfer functions are reduced to scalar transfer functions. Furthermore, by choosing the actual state (ω_z) to be identical to the disturbance (also ω_z), the body dynamics and the internal model of body dynamics simplify to unity.

As an example, it is assumed that the sensory dynamics ($SCC(s)$) may be represented by an high-pass filter with a cut-off frequency equal to the inverse of the dominant time constant of the sensory afference:

$$SCC(s) = \frac{\tau s}{\tau s + 1}$$

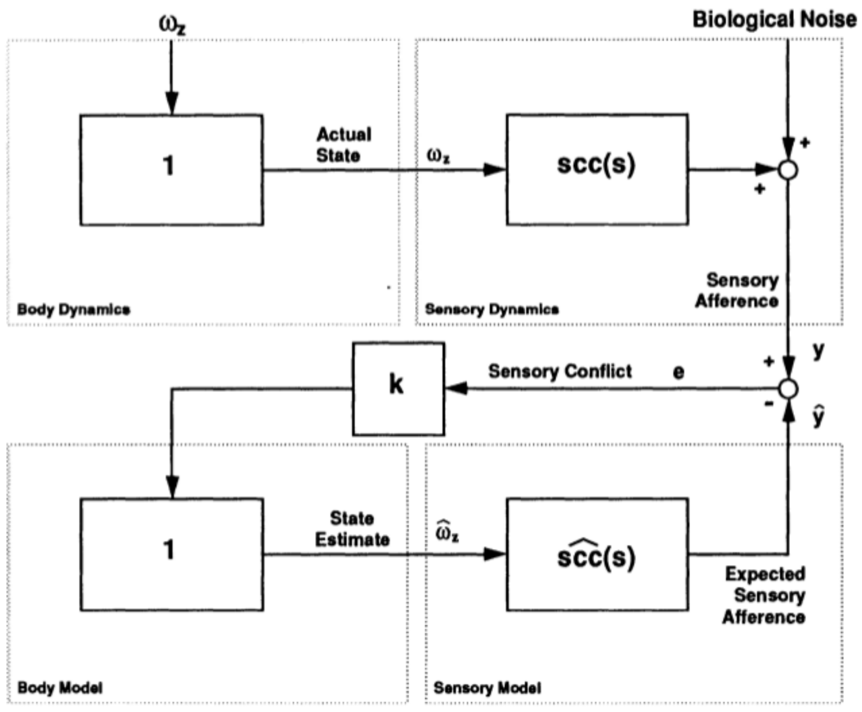


FIGURE 5.2: Simplified one-dimensional sensory conflict model.

It is also assumed that the internal model of sensory dynamics and the real sensory dynamics have a similar form. Therefore:

$$S\hat{C}C(s) = \frac{\hat{\tau}s}{\hat{\tau}s + 1}$$

Using algebra, it is possible to find the transfer function between the internal estimate of angular velocity ($\hat{\omega}_z$) and the actual angular velocity (ω_z). The transfer function has the form:

$$\frac{\hat{\omega}_z}{\omega_z} = \frac{k\ \tau s(\hat{\tau}s + 1)}{((k+1)\hat{\tau}s + 1)(\tau s + 1)}$$

Goldberg and Fernandez [24] estimated that the dominant time constant of the semicircular canals is 5.7 seconds. If the afferent response is modelled with a time constant of 5.7 seconds and also the internal model has a time constant of 5.7 seconds, a pole zero cancellation is obtained. This yields the transfer function

$$\frac{\hat{\omega}_z}{\omega_z} = \frac{k\ \tau s}{(k+1)(\tau s + 1)}$$

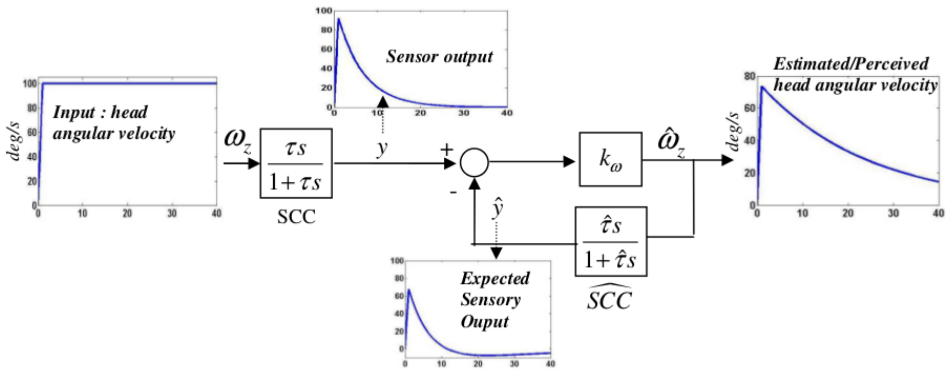

Input : head angular velocity
deg/s
ω_z
$\frac{\tau s}{1+\tau s}$
SCC
Sensor output
y
+
-
$\hat{y}$
Expected Sensory Ouput
k_ω
$\hat{\omega}_z$
$\frac{\hat{\tau} s}{1+\hat{\tau} s}$
$\widehat{SCC}$
Estimated/Perceived head angular velocity
deg/s

FIGURE 5.3: Merfeld's observer model for a yaw rotation.

Figure 5.3 shows the response of this model to a trapezoidal step of yaw angular velocity. The response is observed to have a dominant time constant of:

$$\tau' = (k+1)\tau s$$

5.2 Three-dimensional model

The one-dimensional model is going to be extended to a three-dimensional representation by replacing all scalar values (ω_z, $\hat{\omega}_z$, y, $\hat{y}$ and e) with vectors having three components ($\overrightarrow{\omega}$, $\overrightarrow{\hat{\omega}}$, $\overrightarrow{y}$, $\overrightarrow{\hat{y}}$ and $\overrightarrow{e}$), by replacing the unity operators of the scalar model with 3×3 identity matrices (I), and by replacing the transfer function of the semicircular canal with a 3×3 matrix transfer function. For simplicity, let:

$$S_{scc}(s) = \begin{bmatrix} scc(s) & 0 & 0 \\ 0 & scc(s) & 0 \\ 0 & 0 & scc(s) \end{bmatrix}$$

Through the use of this diagonal matrix transfer function, it is implicitly assumed that the semicircular canals are mutually orthogonal and that they are aligned with the axes of the coordinate system. As a first approximation, the canals may be treated as orthogonal, and the coordinate system may be chosen to align with that defined by the semicircular canals. Anatomical accuracy may easily be provided by changing the transfer function matrix from the diagonal form shown above to one which truly represents the geometry of the semicircular canals. The feedback gain matrix (K) will require analogous changes to represent the change in the sensory matrix. From an input/output perspective, however, these additions are transparent, and therefore are avoided. Since simplicity is gained without any cost in terms of performance, the simplest possible representation has been chosen.

Three-dimensional rotation will, in general, constantly change the orientation of the gravitational force. If the current position of gravity ($\vec{g}_0$) is known, and there is an imposed angular disturbance ($\vec{\omega}$), it is easy to keep track of the orientation of gravity. This physical effect is represented as part of the body dynamics, shown in Figure 5.4.

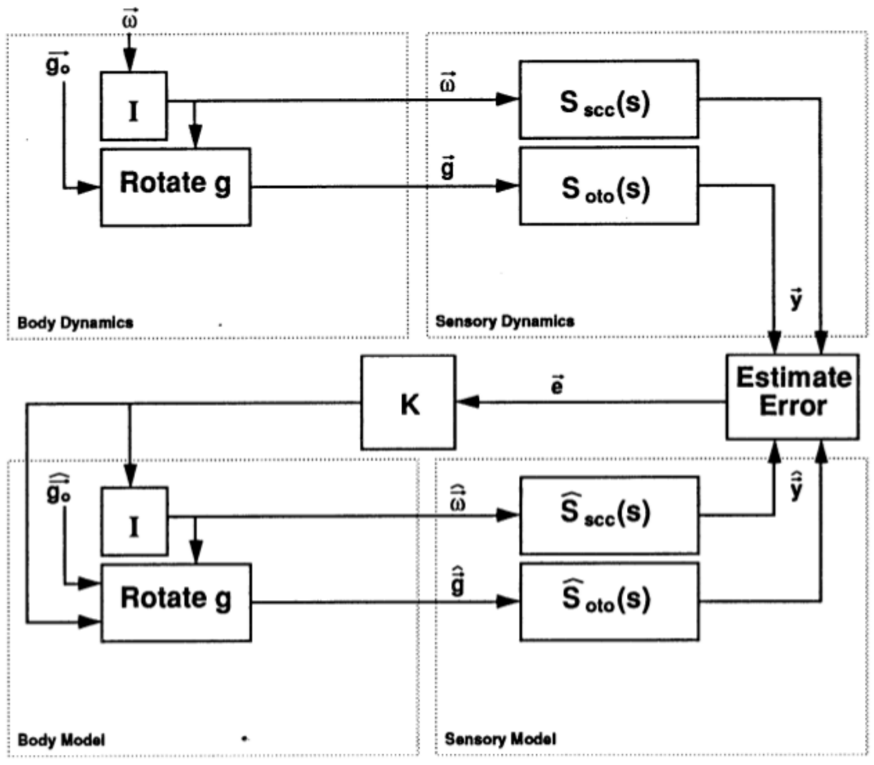


FIGURE 5.4: Three-dimensional sensory conflict model.

The block labelled "rotate g" performs these calculations. The actual calculation are implemented via a quaternion integration.

Analogously, if there is a current internal estimate of gravity, and there is an internal estimate of angular velocity ($\hat{\vec{\omega}}$), it is possible to keep track of an estimate of the orientation of gravity ($\hat{\vec{g}}$) through the same calculations as discussed above. The block "rotate g" is also used to perform these calculations.

By arguments similar to those used when choosing the transfer function representing the semicircular canals, the otolith organs are represented with a diagonal transfer function:

$$S_{oto}(s) = \begin{bmatrix} oto(s) & 0 & 0 \\ 0 & oto(s) & 0 \\ 0 & 0 & oto(s) \end{bmatrix}$$

This representation of the otolith organs assumes that the sensory afference from the two otolith organs, the utricle and the saccule, is integrated to yield a three-dimensional representation of the gravito-inertial force.

Fernandez and Goldberg [25], investigated the dynamics response of first order otolith afferents. They found that the frequency response of the regular units is approximately constant to about 2Hz. Since the chosen inputs for the tests (see Section 8.1) are limited to less than 1 Hz, the otolith transfer function may be approximated as unity:

$$oto(s) = 1.$$

Therefore:

$$S_{oto}(s) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Data at higher frequencies would be required to determine a more accurate transfer function representation.

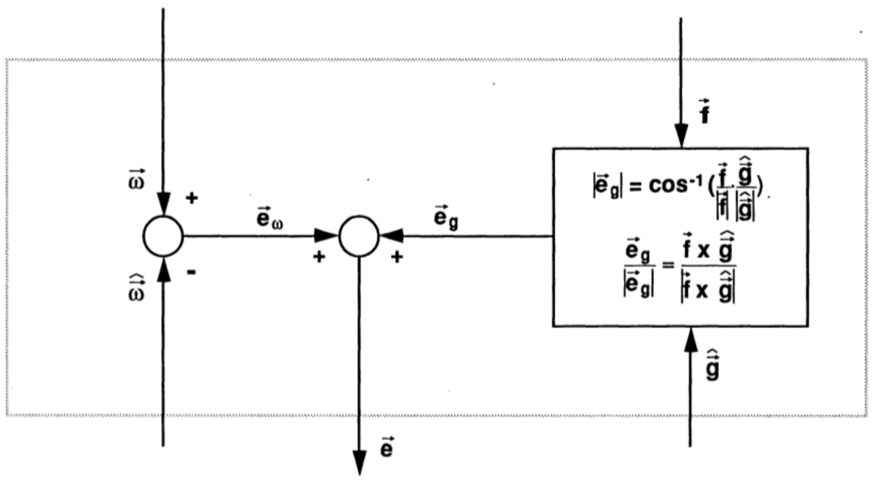
The quaternion integration which keeps track of the orientation of gravity is, obviously, a non-linear calculation. Therefore, error estimation should not be limited to subtraction as previously exhibited for linear systems.

Figure 5.5 shows the equations which determine the six-component error vector ($\overrightarrow{e}$). The linear portion of the process is estimated using subtraction:

$$\overrightarrow{e}_\omega = \overrightarrow{\omega} - \hat{\overrightarrow{\omega}}$$

The estimation of the non-linear part of the error is more complicated. The magnitude of the error equals the angle between the specific force vector ($\overrightarrow{f}$) and the current internal estimate of the GIF ($\hat{\overrightarrow{f}}$). Mathematically, this is written:

$$|\overrightarrow{e}_f| = \arccos\left(\frac{\overrightarrow{f} \cdot \hat{\overrightarrow{f}}}{||\overrightarrow{f}|| \; ||\hat{\overrightarrow{f}}||}\right)$$



$\vec{f}$
$|\vec{e}_g| = \cos^{-1}\left(\frac{\vec{f}\cdot\hat{\vec{g}}}{|\vec{f}|\,|\hat{\vec{g}}|}\right)$
$\frac{\vec{e}_g}{|\vec{e}_g|} = \frac{\vec{f}\times\hat{\vec{g}}}{|\vec{f}\times\hat{\vec{g}}|}$
$\vec{\omega}$
$\hat{\vec{\omega}}$
$\vec{e}_\omega$
$\vec{e}_g$
$\hat{\vec{g}}$
$\vec{e}$
+
-
+
+

FIGURE 5.5: Three-dimensional error computation.

The direction of the gravitational error vector is given by the direction of the rotation needed to align the total gravito-inertial force ($\vec{f}$) with the internal estimate of the specific force ($\hat{\vec{f}}$). Alternatively, this direction can be considered as the direction of the rotation which could yield the discrepancy between the actual and the estimated gravito-inertial force. With either interpretation, this direction may be written as

$$\frac{\vec{e}_f}{|\vec{e}_f|} = \frac{\vec{f}\times\hat{\vec{f}}}{||\vec{f}\times\hat{\vec{f}}||}$$

The total error vector ($\vec{e}$) is formed by summing the the error vectors such that the angular velocity error ($\vec{e}_\omega$) fills the first three components of the error vector, and the gravitational error ($\vec{e}_f$) completes the bottom three elements of the error vector. This may be written:

$$\vec{e} = \begin{bmatrix} \vec{e}_\omega \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vec{e}_f \end{bmatrix}$$

The 6×6 gain matrix (K) is chosen to have just nine non-zero elements as shown below:

$$K = \begin{bmatrix} k_{\omega_x} & 0 & 0 & & & \\ 0 & k_{\omega_y} & 0 & & 0 & \\ 0 & 0 & k_{\omega_z} & & & \\ k_{f\omega_x} & 0 & 0 & k_{f_x} & 0 & 0 \\ 0 & k_{f\omega_y} & 0 & 0 & k_{f_y} & 0 \\ 0 & 0 & k_{f\omega_z} & 0 & 0 & k_{f_z} \end{bmatrix}$$

The angular velocity error feedback gains (k_{ω_x}, k_{ω_y} and k_{ω_z}) are set such that the appropriate velocity storage is obtained for each independent axis of rotation.

From physics it is known that gravito-inertial force is the difference vector formed by subtracting acceleration from gravity or vice versa:

$$\vec{f} = \vec{g} - \vec{a}$$

If an animal (or human) has an internal representation of gravity and a sensory measurement of gravito-inertial force, it is consistent that acceleration might be calculated as the vector difference between these quantities:

$$\hat{\vec{a}} = \hat{\vec{g}} - \hat{\vec{f}}$$

It will be assumed that this is the case.

So, after all these considerations, the final three-dimensional model by Merfeld is represented in Figure 5.6.

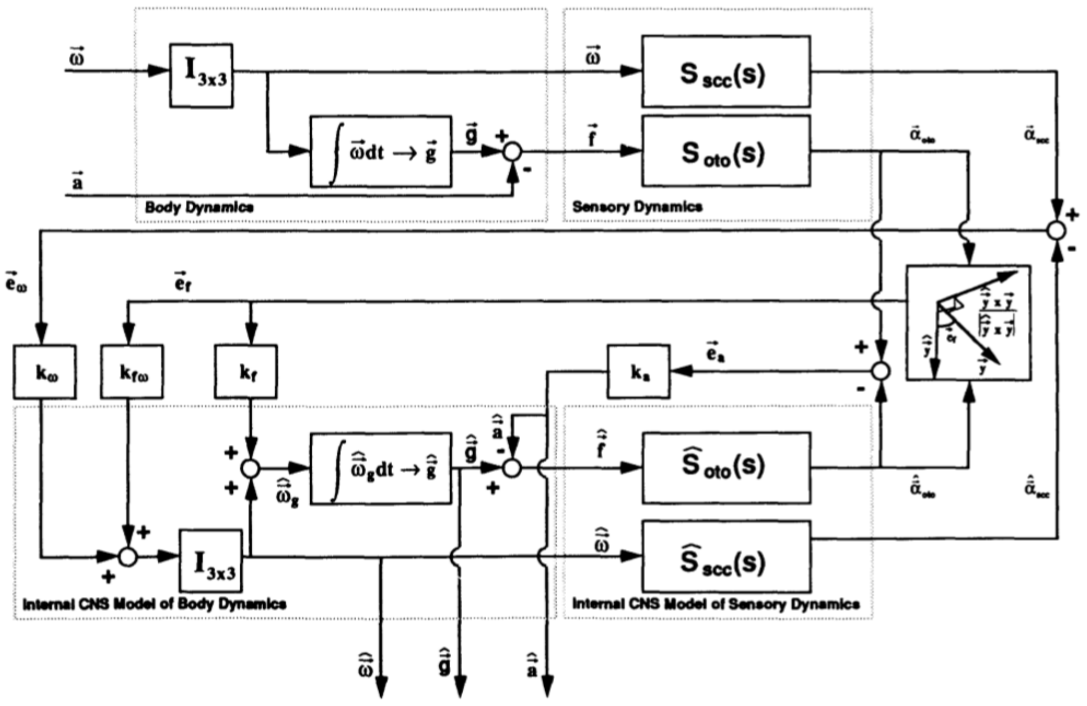


Figure 5.6: Merfeld's three-dimensional model representation.

5.3 Results

The sensory conflict model appears to explain some experimental data. There are, however, a number of obvious improvements which can be made.

This version of the model was developed to investigate and model spatial orientation in the squirrel monkey. It is well known that humans and squirrel monkeys differ dramatically in some of their responses to gravito-inertial force. Changing this model to represent more closely the human sense of spatial orientation is a trivial exercise. The main changes will be in the values of the nine non-zero feedback gains.

Furthermore, more work, experimental and theoretical, needs to be done in order to more accurately represent the sensation of linear motion. The process by which gravito-inertial force is resolved into gravity and linear acceleration is not completely understood and it is a problem which needs to be further investigated.

Vision was completely excluded from playing any role in this model. The importance of vision is, however, very well known, and, therefore, represents an error in the model which needs to be corrected.

Chapter 6

Multi-sensory Observer Model

The observer model developed so far is limited to SCC and otolith cue interaction. The model presented in this chapter starts from the core of Merfeld's sensory conflict model to include additional state estimates together with static and dynamic visual cue inputs. The previous model was able to predict orientation, linear acceleration and angular velocity, but did not predict position in space. To do this, an additional ("limbic") coordinate frame, aligned with the perceived vertical DOF, is added, and velocity and position "path" integration are assumed to take place in this frame. The sole vestibular portion of the model is then tested, and results reproduce stimulus paradigms described in papers by Merfeld & Zupan [26]. Newman et al. [27], [28] performed these tests and then compared the extended model results with the KF model results and original data considered by Borah [29] for the simple visual-vestibular motion paradigms.

6.1 Extended vestibular model

The vestibular core of the observer visual-vestibular interaction model (see Figure 6.1), is a modified and extended version of the model proposed by Merfeld and Zupan [26]. The topology of the model, as shown in the figure, has been rearranged to resemble the presentation format of Haslwanter et al. [30] and has been extended to include the additional state estimates (position $\hat{\vec{x}}$ and velocity $\hat{\vec{v}}$) necessary for displacement estimation and visual sensory interaction. To obtain these estimates, it is assumed that the CNS integrates the perceived linear acceleration vector ($\hat{\vec{a}}$) in an world reference frame oriented to the local vertical.

This "limbic coordinate frame" is defined by the quaternion vector ($\hat{\vec{q}}$) from the estimated gravity state ($\hat{\vec{g}}$). At each time step of the simulation, the estimated linear acceleration vector is transformed to the limbic coordinate system and integrated twice to obtain estimates of velocity and position. For a detailed description of the quaternion mathematics and of the transformation methods employed, it is possible to refer to Appendix A.

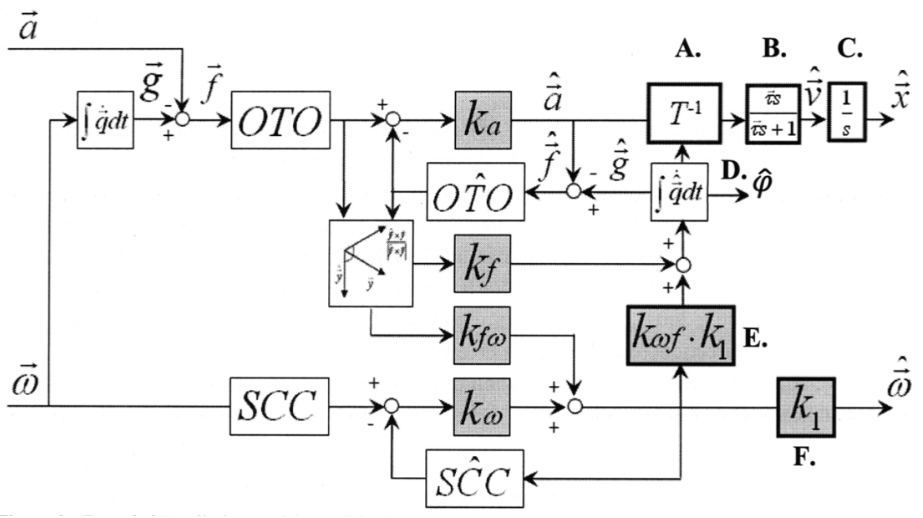


FIGURE 6.1: Extended visual-vestibular model. Modifications to the original Merfeld & Zupan [26] model are outlined in black and denoted A - F. (A) Head to limbic coordinate frame transformation. (B) Leaky integrator for velocity estimate. (C) Integrator for position estimate. (D) Estimated azimuth (E-F). Additional feedback gains.

The estimated quaternion vector ($\vec{\hat{q}}$) also defines perceived azimuth ($\overrightarrow{hatphi}$) within the limbic coordinate frame. Azimuth is an important physical estimate with implications in both laboratory and real world orientation/navigation simulations. Previous observer model implementations neglected azimuth calculation. One interesting prediction is the sustained progression of azimuthal angle in response to Off-Vertical Axis Rotation (OVAR). The classic description of an OVAR simulation is a sensation of motion which proceeds along a conical path while facing the same direction (e.g. constant perceived azimuth). The model, however, predicts a sustained, although greatly reduced, sensation of perceived rotation which continually alters the estimated *heading*, or azimuth angle.

The integration of acceleration to get velocity is accomplished with a leaky integrator with individual time constants for motion around each limbic coordinate axes ($\tau' = [16.66, 16.66, 1.0]^T$).

Two additional weighting parameters are added to the model. The first ($k_{\omega f}$) is nominally set to 1.0 and allows the user to control the influence of angular velocity on the rate of change of gravity. The second parameter (k_l) is a function of the angular velocity residual weighting parameter ($k_l = (k_\omega + 1)/k_\omega$) and is required to make the loop gain of the angular velocity feedback loop unitary.

6.2 Visual-vestibular interaction

The vestibular model extensions described in the previous section are essential prerequisites for the addition of visual sensory information. With this modified vestibular model in place, a visual-vestibular sensory interaction model is now proposed.

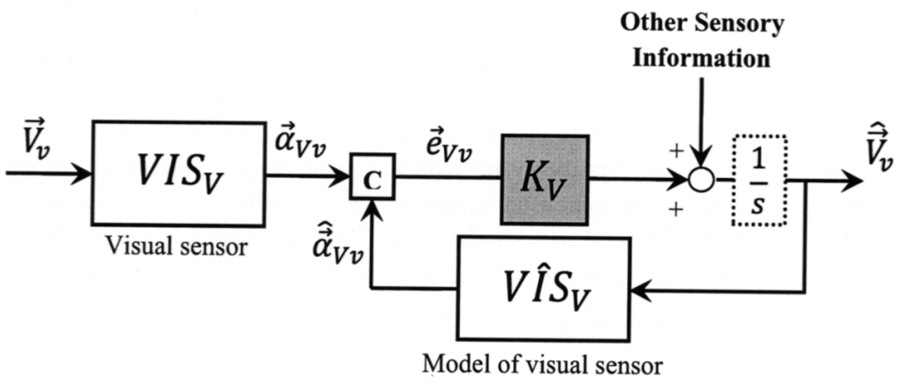


FIGURE 6.2: Representation for a generic visual model pathway.

To keep the structure and notation consistent with the original Merfeld et al. [21] model, each visual pathway is constructed as shown in Figure 6.2. For a generic visual pathway V, a visual input ($\hat{\vec{V}}_v$) is processed by the visual sensor (VIS_V) to generate a visual sensory estimate ($\vec{a}_{V_v}$). This estimate is compared (C) to an expected visual sensory estimate ($\hat{\vec{a}}_{V_v}$) from an internal model of the visual sensor ($V\hat{I}S_V$). The comparative difference ($\vec{e}_{V_v}$) ("sensory conflict") is weighted with a residual weighting parameter (K_V) and added to the rate of change of the estimated state ($\hat{\vec{V}}_v$). In the end, the weighted conflict vector is added to the derivative of the state. Since Merfeld did not include an integrator in the forward loop of the angular velocity feedback pathway, it is added the weighted visual angular velocity error directly to the state itself.

The visual system is assumed to be capable of extracting four visual cues from its environment and these are in the final model represented in Figure 6.3. These are position ($\vec{x}_v$), velocity ($\vec{\dot{x}}_v$), angular velocity ($\vec{\omega}_v$), and gravity ($\vec{g}_v$). The visual input variables are represented by three-dimensional vectors in a right handed, orthogonal, world-fixed, frame of reference (X_W, Y_W, Z_W). To ensure congruency with the observer model's head and limbic coordinate frames, the visual cues are transformed through a rotation matrix (T) to their respective frames of interaction prior to sensory processing. Visual gravity and visual angular velocity are transformed to the head-fixed coordinate axes and visual position and velocity are transformed to the perceived limbic frame, see Appendix A.

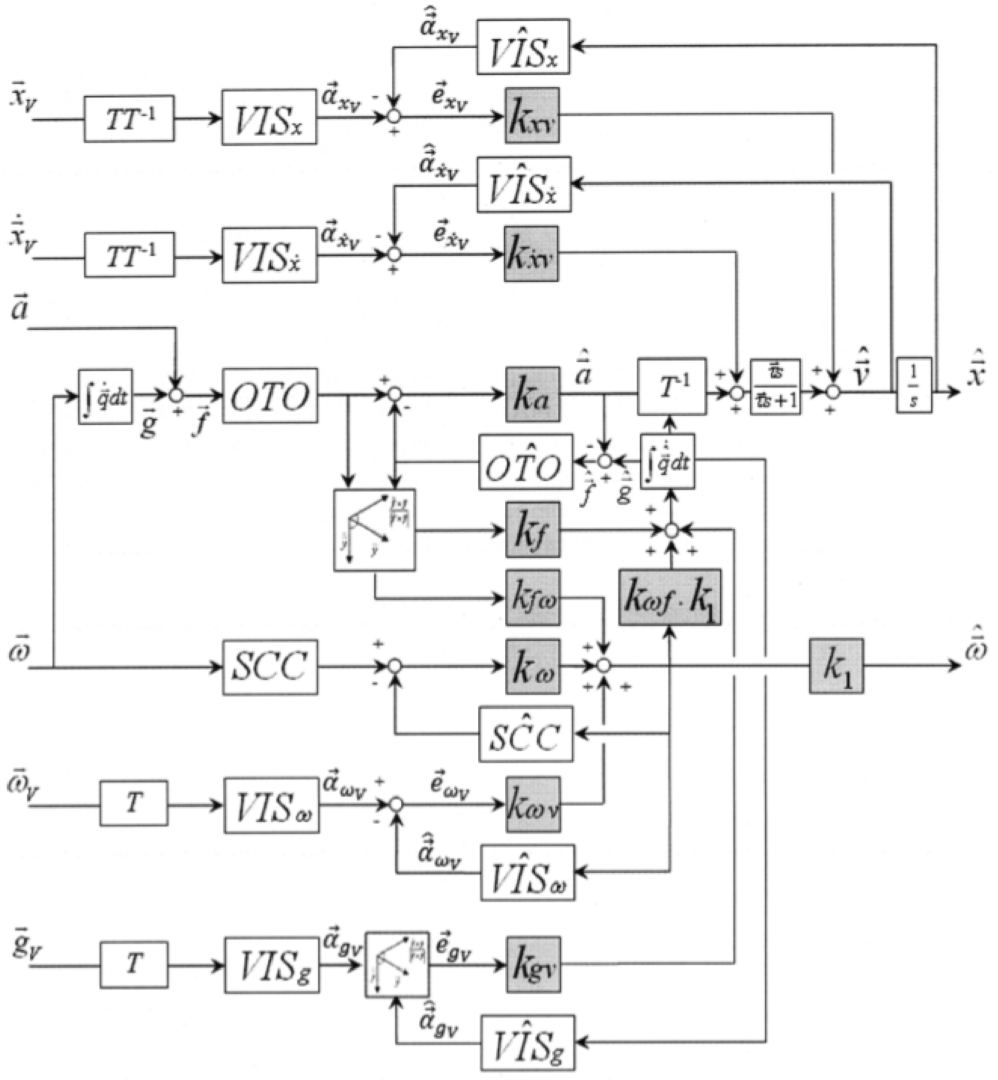


FIGURE 6.3: Visual-vestibular interaction model. Vestibular and visual free parameters are highlighted in gray.

For simplicity, it is assumed that the visual system sensory dynamics can be approximated to identity for both static and dynamic visual inputs. This simplified visual model allows for a baseline assessment of the usefulness and practicity of observer theory for modelling multi-sensory interaction. In three-dimensional space it is possible to represent each visual sensor as a 3×3 identity matrix. Since dynamic inputs illicit a sensation of motion in the opposite direction of the visual field (e.g. linear vection and circular vection), the dynamic sensors are modelled as negative 3×3 identity matrices. Each sensor transforms visual input ($\overrightarrow{x}_v$ $\overrightarrow{\dot{x}}_v$ $\overrightarrow{\omega}_v$ $\overrightarrow{g}_v$) to visual sensory estimates ($\overrightarrow{\alpha}_{x_v}$ $\overrightarrow{\alpha}_{\dot{x}_v}$ $\overrightarrow{\alpha}_{g_v}$ $\overrightarrow{\alpha}_{\omega_v}$).

- $\frac{\overrightarrow{\alpha}_{x_v}}{\overrightarrow{x}_v} = VIS_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_{3\times 3}$
- $\frac{\overrightarrow{\alpha}_{\dot{x}_v}}{\overrightarrow{\dot{x}}_v} = VIS_{\dot{x}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = -I_{3\times 3}$

- $\frac{\overrightarrow{\alpha}_{g_v}}{\overrightarrow{g}_v} = VIS_g = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_{3\times 3}$
- $\frac{\overrightarrow{\alpha}_{\omega_v}}{\overrightarrow{\omega}_v} = VIS_{\omega} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = -I_{3\times 3}$

The next step is the assumption that the CNS has accurate internal models for each visual sensor. Since the CNS already accounts for the proper direction of the visual estimate, all internal models of visual sensory dynamics can be represented as positive 3×3 identity matrices. The internal model of the visual sensors transforms the central state estimates ($\hat{\overrightarrow{x}}_v$ $\widehat{\overrightarrow{\dot{x}}}_v$ $\hat{\overrightarrow{g}}_v$ $\hat{\overrightarrow{\omega}}_v$) to expected visual sensory estimates ($\hat{\overrightarrow{\alpha}}_{x_v}$ $\hat{\overrightarrow{\alpha}}_{\dot{x}_v}$ $\hat{\overrightarrow{\alpha}}_{g_v}$ $\hat{\overrightarrow{\alpha}}_{\omega_v}$).

- $\frac{\hat{\overrightarrow{\alpha}}_{x_v}}{\hat{\overrightarrow{x}}_v} = V\hat{I}S_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_{3\times 3}$
- $\frac{\hat{\overrightarrow{\alpha}}_{\dot{x}_v}}{\widehat{\overrightarrow{\dot{x}}}_v} = V\hat{I}S_{\dot{x}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = -I_{3\times 3}$

- $\frac{\hat{\overrightarrow{\alpha}}_{g_v}}{\hat{\overrightarrow{g}}_v} = V\hat{I}S_g = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_{3\times 3}$
- $\frac{\hat{\overrightarrow{\alpha}}_{\omega_v}}{\hat{\overrightarrow{\omega}}_v} = V\hat{I}S_{\omega} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_{3\times 3}$

A sensory conflict vector is calculated for each visual input based on the relative error between the actual and expected visual sensory estimates. The visual position, velocity and angular velocity errors are calculated through vector subtraction. Each error is represented as a vector containing an individual sensory conflict for each orthogonal axis.

$$\overrightarrow{e}_{x_v} = \overrightarrow{\alpha}_{x_v} - \hat{\overrightarrow{\alpha}}_{x_v}$$

$$\vec{e}_{\dot{x}_v} = \vec{\alpha}_{\dot{x}_v} - \hat{\vec{\alpha}}_{\dot{x}_v}$$

$$\vec{e}_{\omega_v} = \vec{\alpha}_{\omega_v} - \hat{\vec{\alpha}}_{\omega_v}$$

The gravitational error requires both a magnitude and directional component. The conflict vector between the actual and expected gravitational sensory estimates is calculated by computing the rotation required to align both vectors. For the directional component, it is used a cross product to calculate a unit vector perpendicular to the plane formed by the two vectors.

$$\frac{\vec{e}_{g_v}}{|\vec{e}_{g_v}|} = \frac{\vec{\alpha}_{g_v} \times \hat{\vec{\alpha}}_{g_v}}{||\vec{\alpha}_{g_v} \times \hat{\vec{\alpha}}_{g_v}||}$$

For the magnitude, it is used a dot product to calculate the angle required to align both vectors within the previously calculated plane. Note that this implementation is identical to Merfeld's GIF rotational error.

$$|\vec{e}_{g_v}| = \arccos\left(\frac{\vec{\alpha}_{g_v} \cdot \hat{\vec{\alpha}}_{g_v}}{||\vec{\alpha}_{g_v}||\ ||\hat{\vec{\alpha}}_{g_v}||}\right)$$

Then, the error signals are individually weighted with residual weighting parameters that can be adjusted by the user to fit data. The visual gravity residual weighting parameter (k_{g_v}) determines the influence of the visual gravitational error ($\vec{e}_{g_v}$) on the rate of change of the internal estimate of gravity ($\hat{\vec{g}}$). The visual angular velocity residual weighting parameter (k_{ω_v}) determines the influence of the visual angular velocity error ($\vec{e}_{\omega_v}$) on the internal estimate of angular velocity ($\hat{\vec{\omega}}$). The visual position residual weighting parameter (k_{x_v}) determines the influence of the visual position error ($\vec{e}_{x_v}$) on the rate of change of the internal estimate of position ($\hat{\vec{x}}$). The visual velocity residual weighting parameter ($k_{\dot{x}_v}$) determines the influence of the visual velocity error ($\vec{e}_{\dot{x}_v}$) on the rate of change of the internal estimate of velocity ($\hat{\vec{v}}$).

6.3 Results

With all these extensions in place, the modified vestibular model was validated by Newman [27] [28] against a large set of classic vestibular motion paradigms:

- linear and angular acceleration steps
- post-rotatory tilt
- constant velocity Earth vertical yaw rotation
- forward linear acceleration on a sled

- fixed and variable radius centrifugation
- static and dynamic roll tilt
- off-vertical-axis rotation (OVAR)
- large amplitude horizontal and vertical sinusoidal displacements.

The model predicts accurate perceptual responses for the experimental simulations considered and validated by Merfeld [20], [21], [26]. In contrast to the OVAR motion characteristics described by Wood et al [31], the model predicts a sustained progression of perceived azimuthal direction, or heading. The model is also able to take into account the large phase and magnitude estimation errors witnessed experimentally (Jones et al [32]) for large amplitude, low frequency vertical motion. The model predicts a sustained sensation of rotation in the light and a gradual onset of circular vection sensation in response to pure optokinetic drum rotation. The model predicts that the "pitch up" illusion during forward linear acceleration in the light is suppressed due to visual gravity/"down" cues and not visual linear vection information.

Chapter 7

The Final model

As seen in previous chapters, there are a lot of perception models that have been used to describe human spatial orientation. However, the aim of this work is to obtain a good Motion Cueing Algorithm, so it is necessary to analyse and to test existing models to find out which one is better for this purpose. The chosen one will be the starting point and it will be developed until reaching the final version on which will be based the MPC Motion Cueing Algorithm.

7.1 Choice of the model

First of all, Merfeld's model [21] can be considered a good starting point. In particular, it was demonstrated that both the one-dimensional and the three-dimensional models describe in a good way the experimental test done on squirrel monkeys. So, it is a good idea to test if this model can be applied to humans. Clearly, the driving simulator problem needs the three-dimensional model because a realistic description of the 3D space is needed to reproduce realistic feelings. Three dimensions imply a more complicated model, thus the computational burden will be higher: the calculation of the direction error $\overrightarrow{e_f}$, in fact, involves a cross product, an inverse cosine function and some euclidean norms that clearly elevate the cost of the calculations. Performances and computational cost are very important for the purpose of this thesis, so these aspects will be always taken into consideration and will be discussed later.

The other model analysed in Chapter 6, is the multi-sensory observer model by Newman [28]. It is based on an extended Merfeld model, see [26], that involves also the estimation of linear position and linear velocity using visual cues and some leaky integrators.

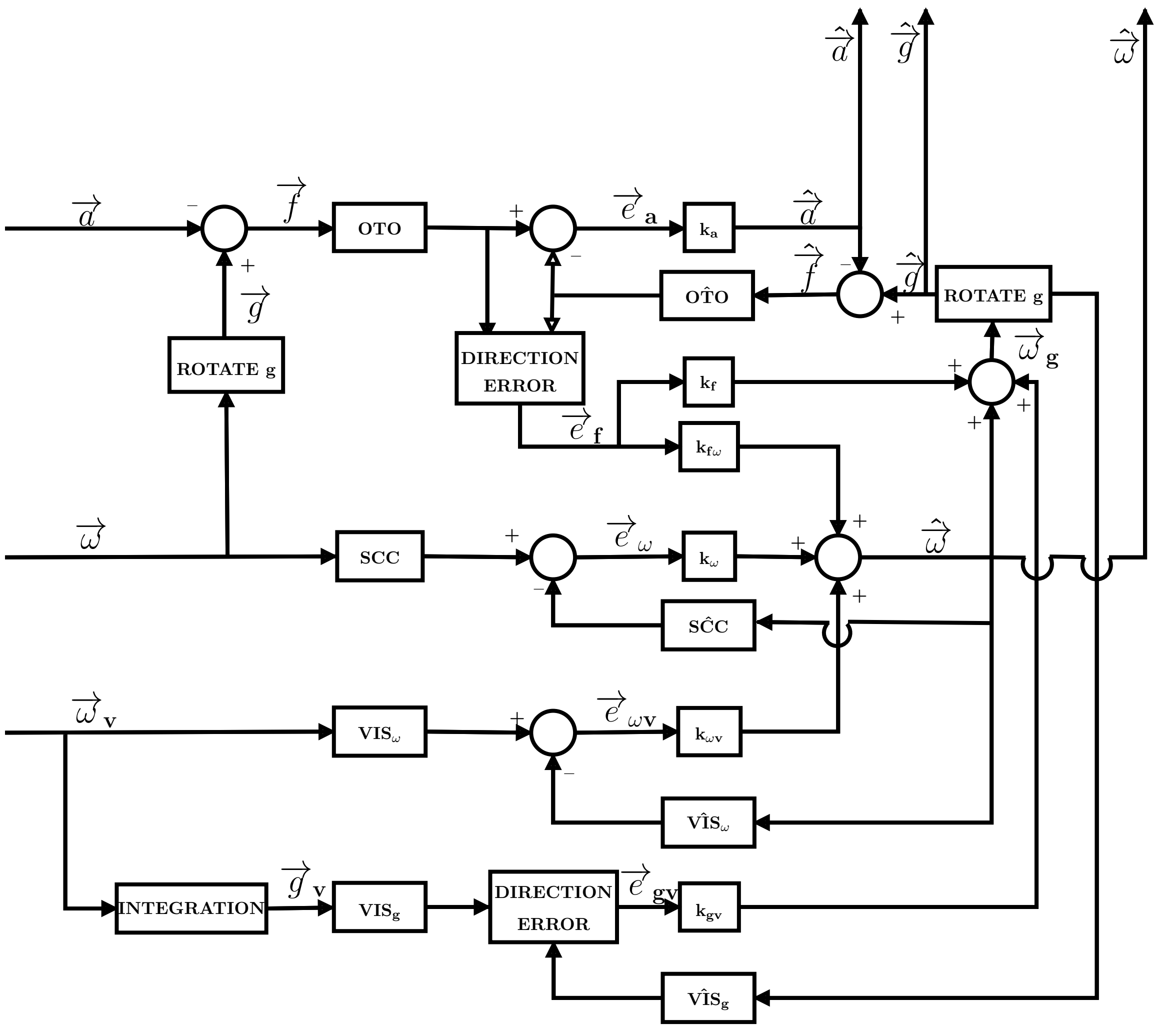


FIGURE 7.1: Final model chosen for this work.

The model is called "multi-sensory" because dynamic and static visual cues are introduced. In particular, it is added visual information about angular velocity and gravity perception, such that the driver can feel the rotations of the graphics.

It is clear that these models become more and more complicated. To simplify Newman's model, visual parts regarding linear position and linear velocity are excluded while angular velocity and gravity perception are maintained. These approximations are made to obtain a model as simple as possible to keep the computational cost limited, while obtaining results that are detailed enough to describe accurately realistic dynamics. There is a clear trade-off between these two aspects. The model used in this thesis lays in the middle point and tries to obtain good performances both in the results (i.e. good tracking) and in the computational cost (fast simulation time).

The final model is represented in Figure 7.1, all its features will be presented in the next sections.

7.1.1 Motivations

First of all, we started from Merfeld's model that showed it could well replicate mammals (probably also humans) cues, but it lacks something. It presumes that the driver does not have any visual information during the test, such as he is blind or the test is performed into the darkness, and this is not a realistic situation for a driving simulation session. More, one purpose of this thesis is to track references of linear acceleration and angular velocity, not the ones of the car during the circuit lap, but the ones perceived by the driver's CNS. This task can not be achieved in a realistic way without using visual cues, in fact, the natural way the model uses to follow a fast acceleration change is to apply the tilt coordination by tilting the platform, but this behaviour modifies also the driver's gravity perception $\hat{\vec{g}}$. This behaviour is an implicit feature of the sensory conflict model and it can not be avoided. But the driver must not feel tilted, so the model formulation needs a revision that includes a new input (control) which can keep the estimated gravity vector at acceptable values even if tilt coordination is applied.

This is done with the control of the visual angular velocity and visual gravity as described in Figure 7.1. In this way, in fact, the estimated gravity is driven to the right value and the controls of linear acceleration and angular velocity can do their best to track the references.

7.2 Model implementation

Once chosen the final model that is going to be used in this work, another choice must be done: how the model has to be implemented and how it can be tested, namely which tool or programming language has to be used.

The `ACADO toolkit` seemed to be the best available choice to achieve these purposes. It is an open-source tool that deals with a lot of control and optimization applications (see Appendix B). In particular, it works using a set of differential equations that describes the dynamics of the model, so the whole system must be expressed in this way. This rearrangement process is experimental, in fact, this type of model was thought to work in different simulation environment, such as Simulink, in which it is easy to replicate these block schemes. Other approaches to the motion cueing problem (e.g. linear MPC, see Chapter 9) used, for example, a state space representation to describe the dynamics of the system, but this method does not look convenient in this case. So, using a new tool that works in a different way, represents an undiscovered field that can take advantages like disadvantages during the development phase.

7.2.1 Dynamical systems

In the final model there are only two different transfer functions: one representing the SCC dynamic and one for the OTO. Both of them have to be translated from the frequency to the time domain, but fortunately, thank to some simplifications, the transfer function of the otolith organs is simply the unity and it needs no work. It is used a first order transfer function to represent the dynamic of the perceived angular velocity through the SCC:

$$SCC(s) = \frac{\tau s}{\tau s + 1} = 1 - \frac{1}{\tau s + 1}$$

where τ is the time constant of the SCC (5.7s, the same as in Merfeld's paper [21]). Then, it is possible to write the dynamic of the perceived angular velocities in the frequency domain as:

$$\overrightarrow{A}_\omega(s) = SCC(s)\overrightarrow{\Omega}(s) = \overrightarrow{\Omega}(s) - \frac{1}{\tau s + 1}\overrightarrow{\Omega}(s)$$

where $\overrightarrow{A}_\omega(s)$ and $\overrightarrow{\Omega}(s)$ are respectively the Laplace transform of the perceived angular velocity and of the effective angular velocity. Then, passing to the inverse Laplace transform:

$$\mathcal{L}^{-1}\{\overrightarrow{A}_\omega(s)\} = \mathcal{L}^{-1}\{\overrightarrow{\Omega}(s)\} + \overrightarrow{\alpha}^*_\omega$$

where $\overrightarrow{\alpha}^*_\omega = \mathcal{L}^{-1}\{\frac{-1}{\tau s+1}\overrightarrow{\Omega}(s)\}$. To obtain the inverse Laplace transform of $\overrightarrow{A}_\omega(s)$, it is necessary to come back to the time domain and this process can simply be done by transforming both its components singularly thank to the linearity of the inverse Laplace transform. The first piece $\mathcal{L}^{-1}\{\overrightarrow{\Omega}(s)\}$ is just composed by the effective angular velocity ($\overrightarrow{\omega}$), the second piece has a more complex dynamic that can be described through a simple differential equation:

$$\dot{\overrightarrow{\alpha}}^*_\omega = -\frac{\overrightarrow{\omega} + \overrightarrow{\alpha}^*_\omega}{\tau}$$

In the end, it is possible to write the dynamic of the perceived angular velocity through the SCCs with a sum, as:

$$\overrightarrow{\alpha}_\omega = \overrightarrow{\omega} + \overrightarrow{\alpha}^*_\omega$$

Exactly the same deductions can be made for the estimated perceived angular velocity, ($\hat{\overrightarrow{\alpha}}_\omega$), obtaining:

$$\hat{\overrightarrow{\alpha}}_\omega = \hat{\overrightarrow{\omega}} + \hat{\overrightarrow{\alpha}}^*_\omega$$

where $\hat{\overrightarrow{\omega}}$ is the estimated angular velocity and $\hat{\overrightarrow{\alpha}}^*_\omega$ is described by the differential equation:

$$\widehat{\dot{\overrightarrow{\alpha}}}^*_\omega = -\frac{\hat{\overrightarrow{\omega}} + \hat{\overrightarrow{\alpha}}^*_\omega}{\tau}$$

7.2.2 Gravity vector orientation

A key feature of this work lays in the update of the gravity vector orientation $\overrightarrow{g}$. When the driver, sitting on the cockpit, is undergone some rotations, the gravity vector is no more parallel to the z-axis of the driver coordinate frame. The angles that describe this rotation have to be stored to update the orientation of the gravity vector because it has an important role in the dynamics of the model. It is possible to replicate the same reasoning also for the estimated gravity vector $\overrightarrow{\hat{g}}$.

In the previous works [21], [28], the update was made through rotation matrices, using Euler angles or quaternion integration. These methods are equivalent but they both have some lacks: especially, the computational cost is very high because they both need some heavy calculus.

In this work a different implementation approach is used. It is thought principally to achieve good performances that will be demonstrated to be way better than the methods cited before. The problem is to find a good representation of the update of the gravity vector without making the model too complex. So, the idea is to consider the gravity direction as an effective vector that rotates due to an applied angular velocity that can be described by a vector too. In particular, it is possible to describe the dynamic of a rotating vector simply through a set of differential equations, in this way, the dynamic is already expressed in the right way to work in the `ACADO` environment. So, the complexity of the model does not increase, but there are only a few more differential equations to solve.

Now, let see how this update works according to [33]. Consider Figure 7.2, where: A is an arbitrary vector, A' is vector A rotated to a new orientation an infinitesimally short time later ($\Delta t \to 0$), ΔA is the difference between vector A' and vector A (as $\Delta t \to 0$), w is the angular velocity which "rotates" vector A (this is also a vector).

The purpose is to find an expression for $\frac{dA}{dt}$ at the given instant. Since the vector A is physically "rotating" due to w, then (as a result) vector $\frac{dA}{dt}$ is perpendicular to A and w. So, it is possible to write:

$$\frac{d\overrightarrow{A}}{dt} = \overrightarrow{w} \times \overrightarrow{A}$$

Hence, using the vector cross product, a very useful formula relating the derivative of a vector of fixed length to the angular velocity that "rotates" this vector in three-dimensional space is obtained. Note that the above formula also applies if vector A is translating as well as rotating. This means that only a rotation can change the orientation of a vector of fixed length, instead, a translation does not have any influence on its direction.

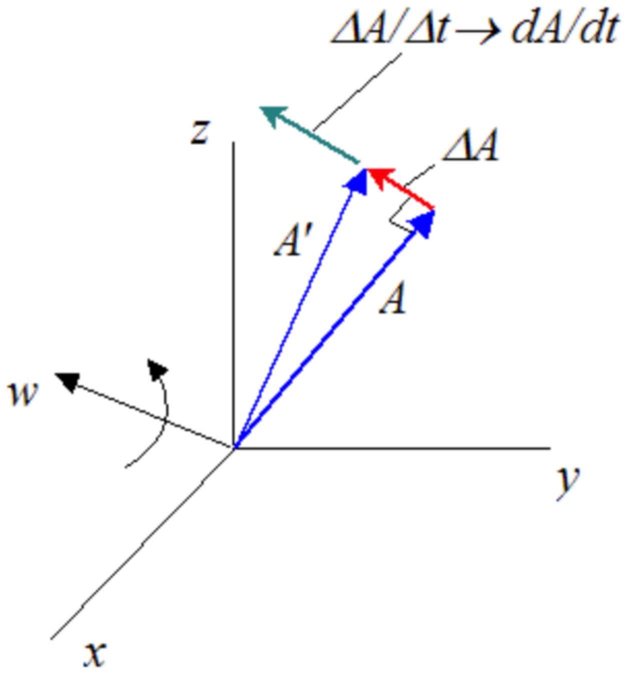


FIGURE 7.2: Fixed length vector derivative.

In the particular case of the model in analysis, it is not really the gravity vector that rotates due to an applied angular velocity, but it is the driver frame that rotates. In fact, the effective gravity vector is fixed in the world coordinate frame and is always parallel to its z axis. So the final relation that describes the update of the gravity vector orientation in the head-fixed driver coordinate frame is:

$$\frac{d\vec{g}}{dt} = -\vec{\omega} \times \vec{g}$$

And, exactly in the same way, for the estimated gravity vector:

$$\frac{d\hat{\vec{g}}}{dt} = -\vec{\omega}_g \times \hat{\vec{g}}$$

7.3 Differential equations of the model

The final model deals with the dynamics of the platform in a three-dimensional space and this representation needs 27 differential states and a lot of intermediate states to be described in the simplest manner. The control needs 9 variables: three for the linear acceleration, three for the angular velocity, three for the visual angular velocity. As in previous models, all these controls are expressed in the driver coordinate frame, so, in the end, it is necessary to perform a transformation (a quaternion integration is used) to obtain all the controls in the world coordinate frame. In this way, the rights values of control can be given to the platform and to the graphics.

The differential states are defined as:

- $x_{1:3} = \overrightarrow{\alpha}^*_\omega$, a part of the perceived angular velocity, see Section 7.2.1
- $x_{4:6} = \hat{\overrightarrow{\alpha}}^*_\omega$, a part of the estimated perceived angular velocity, see Section 7.2.1
- $x_{7:9} = \overrightarrow{g}$, gravity vector
- $x_{10:12} = \hat{\overrightarrow{g}}$, estimated gravity vector
- $x_{13:15} = \overrightarrow{\theta}$, platform pitch, roll and yaw rotation angles
- $x_{16:18} = \overrightarrow{v}$, platform linear velocity
- $x_{19:21} = \overrightarrow{p}$, platform position
- $x_{22:23} = \overrightarrow{\Phi}$, angles for the $\overrightarrow{g}_v$ computation
- $x_{24:27} = \overrightarrow{q}$, quaternion.

And the controls:

- $u_{1:3} = \overrightarrow{\omega}$, angular velocity in the driver coordinate frame
- $u_{4:6} = \overrightarrow{a}$, linear acceleration in the driver coordinate frame
- $u_{7:9} = \overrightarrow{\omega}_v$, visual angular velocity of the graphics in the driver coordinate frame.

7.3.1 Intermediate states

Intermediate states need to be defined, necessary to understand the dynamics and to give the final representation of the model a clear formulation in which all the differential equations are well written.

- the gravito-inertial force $\overrightarrow{f}$, defined as the difference between the gravity vector $\overrightarrow{g}$ and the linear acceleration $\overrightarrow{a}$

$$\overrightarrow{f} = \overrightarrow{g} - \overrightarrow{a}$$

- the estimated gravito-inertial force $\hat{\vec{f}}$, the estimated linear acceleration $\hat{\vec{a}}$ and the gravito-inertial magnitude error $\vec{e}_a$

$$\begin{cases} \hat{\vec{f}} = \hat{\vec{g}} - \hat{\vec{a}} \\ \hat{\vec{a}} = k_a \vec{e}_a \\ \vec{e}_a = \vec{f} - \hat{\vec{f}} \end{cases}$$

  Putting all these formulas together, it is possible to rewrite the estimated gravito-inertial force in a closed-loop way:

$$\Longrightarrow \qquad \hat{\vec{f}} = \frac{\hat{\vec{g}} - k_a \vec{f}}{1 - k_a}$$

- one of the most important state of the model, the gravito-inertial direction error $\vec{e}_f$, that defines the direction error of the gravito-inertial force

$$\vec{e}_f = \left[\frac{\pi}{2} - \arcsin\left(\frac{\vec{f} \cdot \hat{\vec{f}}}{||\vec{f}|| \cdot ||\hat{\vec{f}}||}\right)\right] \frac{\vec{f} \times \hat{\vec{f}}}{||\vec{f} \times \hat{\vec{f}}||}$$

- the estimated angular velocity $\hat{\vec{\omega}}$, the angular velocity magnitude error $\vec{e}_\omega$ and the visual angular velocity magnitude error $\vec{e}_{\omega v}$

$$\begin{cases} \hat{\vec{\omega}} = k_\omega \vec{e}_\omega + k_{f\omega} \vec{e}_f + k_{\omega_V} \vec{e}_{\omega_V} \\ \vec{e}_\omega = \vec{\alpha}_\omega - \hat{\vec{\alpha}} - \omega = \vec{\omega} + \vec{\alpha}^*_\omega - (\hat{\vec{\omega}} + \hat{\vec{\alpha}}^*_\omega) \\ \vec{e}_{\omega_V} = -(\vec{\omega}_V + \hat{\vec{\omega}}_s) \end{cases}$$

  Putting all these formulas together, it is possible to rewrite the estimated angular velocity in a closed-loop way:

$$\Longrightarrow \qquad \hat{\vec{\omega}} = \frac{k_\omega(\vec{\omega} + \vec{\alpha}^*_\omega - \hat{\vec{\alpha}}^*_\omega) + k_{f\omega} \vec{e}_f - k_{\omega_V} \vec{\omega}_V}{1 + k_\omega + k_{\omega_V}}$$

- the estimated gravity-weighting angular velocity $\vec{\omega}_g$ and the visual gravity direction error $\vec{e}_g$

$$\begin{cases} \vec{\omega}_g = \hat{\vec{\omega}} + k_f \vec{e}_f + k_{g_V} \vec{e}_g \\ \vec{e}_g = \left[\frac{\pi}{2} - \arcsin\left(\frac{\vec{g}_v \cdot \hat{\vec{g}}}{||\vec{g}_v|| \cdot ||\hat{\vec{g}}||}\right)\right] \frac{\vec{g}_v \times \hat{\vec{g}}}{||\vec{g}_v \times \hat{\vec{g}}||} \end{cases}$$

Putting all these formulas together, it is possible to rewrite the estimated gravity-weighting angular velocity in a closed-loop way:

$$\Longrightarrow \qquad \overrightarrow{\omega}_g = \frac{k_\omega(\overrightarrow{\omega} + \overrightarrow{\alpha}^*_\omega - \hat{\overrightarrow{\alpha}}^*_\omega) + (k_{f\omega} + k_f(1 + k_\omega + k_{\omega V}))\overrightarrow{e}_f - k_{\omega V}\overrightarrow{\omega}_V + k_{gV}(1 + k_\omega + k_{\omega V})\overrightarrow{e}_g}{1 + k_\omega + k_{\omega V}}$$

- and last, the skew-symmetric angular velocity matrix that is necessary to define the quaternion integration

$$W = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}$$

7.3.2 Equations

Now it possible to simply write down the equations that describe the dynamics of the differential states as defined above. This set of equations is everything the model needs to be described through the `ACADO toolkit`.

$$\begin{cases} \dot{\overrightarrow{\alpha}}^*_\omega = -\frac{\overrightarrow{\omega} + \overrightarrow{\alpha}^*_\omega}{\tau} \\ \dot{\hat{\overrightarrow{\alpha}}}^*_\omega = -\frac{\hat{\overrightarrow{\omega}} + \hat{\overrightarrow{\alpha}}^*_\omega}{\tau} \\ \dot{\overrightarrow{g}} = -\overrightarrow{\omega} \times \overrightarrow{g} \\ \dot{\hat{\overrightarrow{g}}} = -\overrightarrow{\omega}_g \times \hat{\overrightarrow{g}} \\ \dot{\overrightarrow{\theta}} = \overrightarrow{\omega} \\ \dot{\overrightarrow{v}} = \overrightarrow{a} \\ \dot{\overrightarrow{p}} = \overrightarrow{v} \\ \dot{\overrightarrow{\Phi}} = \overrightarrow{\omega}_v \\ \dot{\overrightarrow{q}} = \frac{1}{2}W\overrightarrow{q} + \lambda(1 - q_0^2 - q_1^2 - q_2^2 - q_3^2)\overrightarrow{q} \end{cases}$$

where $\lambda = 0.9$. The quaternion $\overrightarrow{q}$ achieves the storage of all the rotations between the world and the driver coordinate frame. In this way, it is possible to switch from one frame to the other without problems. This mechanism is, in fact, necessary because the control inputs of the model $\overrightarrow{\omega}$, $\overrightarrow{a}$ and $\overrightarrow{\omega}_v$ are fixed in the driver frame but the platform movements must be controlled in the world frame, so this transformation must take place. However, how the quaternion integration works, is better discussed in the Appendix A.

Solving iteratively this set of equations well describes the dynamics of the chosen model and some simulations confirm that it fits the results previously obtained by Merfeld [21] and then by Newman [27].

7.4 Control implementation

Now it is time to set up the MPC, in this case it is a Non-linear Model Predictive Control (NMPC) due to the non-linearities introduced by the model. The purpose of a control system is to generate the right input (control) that permits the output to track some given references. To set up the NMPC the following items are needed:

- output references, in this case the control system wants to focus particularly on the estimated angular velocity $\vec{\hat{\omega}}$, on the estimated linear acceleration $\vec{\hat{a}}$ and on the estimated gravity vector $\vec{\hat{g}}$
- MPC, as said, is an open-loop optimal control, so it needs a cost function, like the one in Section 7.4.1, to be defined
- at the same time, also the weighting matrices associated to the cost function have to be chosen. The weights have a relevant role in optimal control applications and they are going to be discussed later
- limits and bounds of the platform performances must be included, this aspect can simply deal with the chosen MPC approach

Once everything is ready, the control can take place and then it is possible to analyse how it controls the inputs trying to make the outputs track the references.

7.4.1 Cost function, weighting matrices and bounds

As said before, the NMPC solves an optimal control problem which needs a cost function. Mathematically this problem is defined as:

$$\begin{aligned} \min_{\substack{x_0,\dots,x_N \\ u_0,\dots,u_{N-1}}} \quad & \sum_{k=0}^{N-1} ||h(x_k,u_k)-\tilde{y}_k||_W^2 + |||h_N(x_N)-\tilde{y}_N||_{W_N}^2 \\ \text{s.t.} \quad & x_0 = \hat{x}_0 \\ & x_{k+1} = F(x_k,u_k), \ \text{for } k=0,\dots,N-1 \end{aligned}$$

where $F(\cdot)$ describes the discretized system dynamics, previously described.

To define the cost function and the weighting matrices, first it is necessary to decide which variables must be weighted during the control computation. In this way, the reference function is defined as follows:

$$h = [\hat{\vec{\omega}} \quad \hat{\vec{a}} \quad \vec{e}_{\omega} \quad \vec{e}_{a} \quad \vec{e}_{f} \quad \vec{e}_{gv} \quad \vec{e}_{\omega v} \quad \vec{\theta} \quad \vec{p} \quad \vec{v} \quad \vec{\omega} \quad \vec{a} \quad \vec{g}_{v} \quad \vec{\omega}_{v} \quad \hat{\vec{g}}]^T$$

$$h_N = [\vec{x}]$$

where $\vec{x}$ is the vector of the states of the system.

The aim is to follow all the references defined in h, but the most important ones are the estimated linear acceleration, the estimated angular velocity and the estimated gravity vector orientation because the driver perception of motion is achieved with these three estimates. In fact, only these variables will have time-varying references, obtained from real sensors, all the other variables are weighted only to minimize their magnitude. For example, the displacement of the platform $\vec{p}$ from its original/neutral position must be controlled such as there will always be space enough to make the next move. Another example can be done about the error vectors (i.e. $\vec{e}_{\omega}, \vec{e}_{f}, \ldots$), they could be driven to zero, so the CNS estimation of the respective variable is the same as the actual variable value. The controls ($\vec{\omega}$, $\vec{a}$, $\vec{\omega}_{v}$) are weighted too, clearly their weighting parameters will be smaller than other variables to let the controller adjust their values, but it could be useful, in some particular circumstances, to weight more some inputs.

Now, it remains to define the weighting matrices W and W_N. Both the matrices are diagonal, so it is only necessary to set up one value along each axis and for each variable. In the function h totally there are 15 three-dimensional variables to consider, that is 45 values to choose and a 45×45 W matrix to build up. Respectively, for the function h_N there are 27 states, so W_N must be 27×27.

In the end, there are 72 free weighting parameters and this can well give the idea of the complexity of the problem in analysis.

7.4.2 `ACADO` settings

The model is implemented using the `ACADO toolkit`, namely through its Matlab interface. This tool is well built for control and optimization problems (see Appendix B) and has a lot of available settings, so it is interesting to sum up the ones used for the NMPC formulation of this thesis, see Table 7.1, 7.2.

Parameter	Value
INTEGRATOR_TYPE	INT_IRK_RIIA3
NUM_INTEGRATOR_STEPS	1

Table 7.1: ACADO toolkit settings for the model integrator

Parameter	Value
INTEGRATOR_TYPE	INT_IRK_RIIA1
NUM_INTEGRATOR_ STEPS	N
HESSIAN_APPROXIMATION	GAUSS_NEWTON
DISCRETIZATION_TYPE	MULTIPLE_SHOOTING
SPARSE_QP_SOLUTION	FULL_CONDENSING

Table 7.2: ACADO toolkit settings for the NMPC

These parameters are useful and have particular features:

- INTEGRATOR_TYPE, set the integrator type, in particular, the INT_IRK_RIIA1 is a Radau IIA integrator of order 1 (Continuous output Implicit Runge-Kutta)
- NUM_INTEGRATOR_STEPS, set the number of integration steps per call of the integrator
- HESSIAN_APPROXIMATION, set the used approximation for the hessian matrix, in this case it is used the "Gauss-Newton" approximation
- DISCRETIZATION_TYPE, set the way in which the non-linear programming problem is treated. The "MULTIPLE_SHOOTING" option regards system equations as an equality constraint without substituting explicitly, thing that is done by the "SINGLE_SHOOTING" option
- SPARSE_QP_SOLUTION, set the way the QP solver find the solution.

Chapter 8

Simulations and Controls

In this chapter some tests are presented and discussed. They confirm the good implementation of the model and the fact that it can well replicate all the features that its previous versions ([20], [27]) have.

Every test is presented with all its specific features, parameters and simulation time. The performances that are going to be described have been obtained with a commercial desktop computer with Intel i3-4340 3.6GHz CPU and 8GB RAM.

8.1 Simulations

First of all, some simulations are presented. They test some classical experiments that are very important to confirm that a perception model is working in the right way.

In this section, some results from Merfeld's experience [21] are used; the purpose is to build up the sensory conflict model by Merfeld and to verify that the final model, used in this thesis, can replicate all the results that the sensory conflict model produces.

To build the Merfeld's three-dimensional sensory conflict model Simulink is used, see Figure 8.1, that is the nearest and simplest way to replicate his work [20]. In this way, it is possible to perfectly replicate his tests and to compare the obtained results of the Simulink model with the ACADO implementation ones.

8.1.1 Post-rotational tilt

OptoKinetic Nystagmus (OKN) is nystagmus induced by continuous movement of the whole or a part of the visual field, [34]. In monkeys the eyes are powerfully driven by

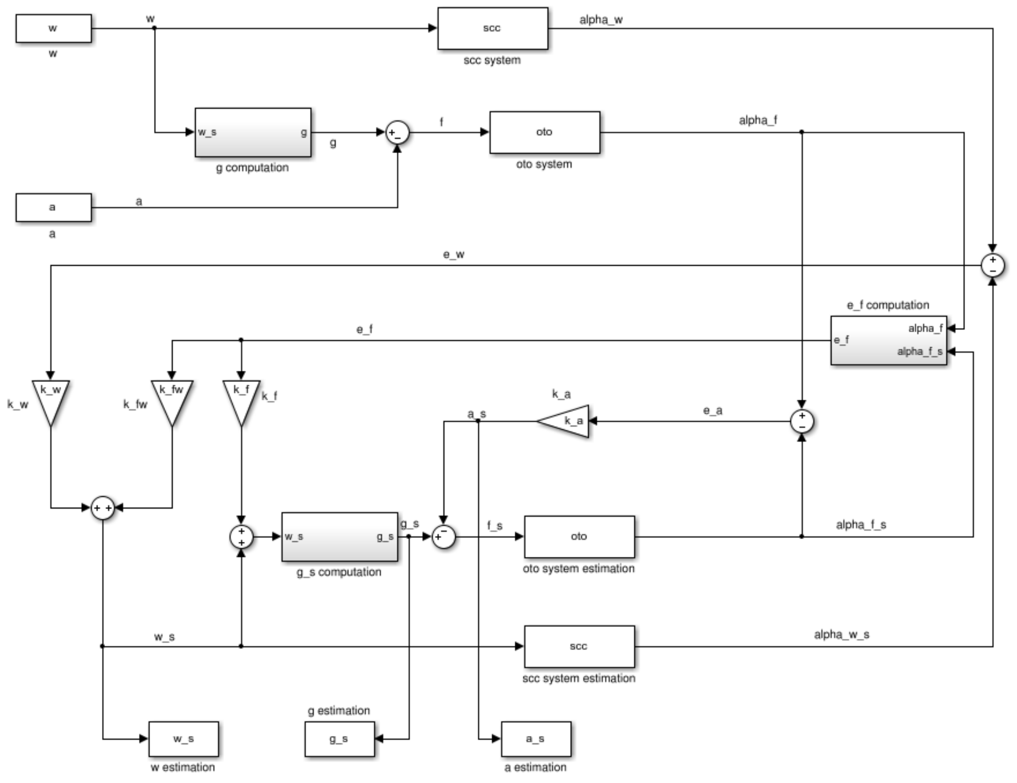


FIGURE 8.1: Simulink sensory conflict model implementation.

whole field rotation and the nystagmus lasts as long as the stimulus continues. When the stimulus is terminated, if animals are in darkness, there is a persistent after-response, OptoKinetic After-Nystagmus (OKAN). In the monkey, appreciable periods of OKAN regularly follow OKN. There are generally two phases of OKAN. In the first phase the nystagmus is in the same direction as during OKN. This is often followed by a second phase which is oppositely directed. See Figure 8.2 for an example of OKN.

FIGURE 8.2: OKN video example.

The post-rotational tilt has to describe the OKAN just illustrated above. So the subject is put into darkness and then he begins to rotate: first a constant yaw velocity of 100°/s is applied and then it is suddenly stopped and a roll rotation of 45° is applied.

Simulink results In Figures 8.3, 8.4, 8.5, it is possible to see the original, Simulink implemented, Merfeld's sensory conflict model results. In these pictures some comparisons between real variables and the estimated ones are represented. Looking at the differences between these two values, it is possible to analyse how the vestibular system reacts to this test:

- only fast angular velocities changes are well perceived, such as they would be filtered by an high-pass filter. Instead, if the velocity is almost constant, the perception of rotation decays with the time constant of the system until the subject feels no more rotation. The OKAN is clearly noticeable at about 50s, when the yaw rotation (red) stops but the subject feels a strong rotation in the opposite direction. Also the perceived pitch angular velocity (green) does not represent faithfully what actually happens: there is a sensation of rotation about the y axis due to the mixed contribute of the false perceived positive yaw velocity and the actual roll rotation

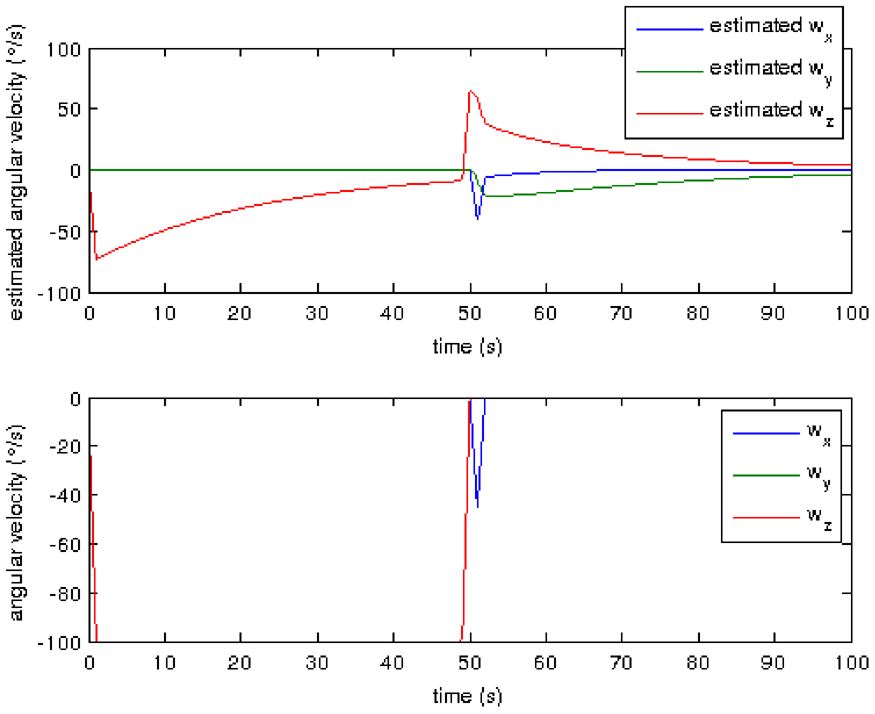
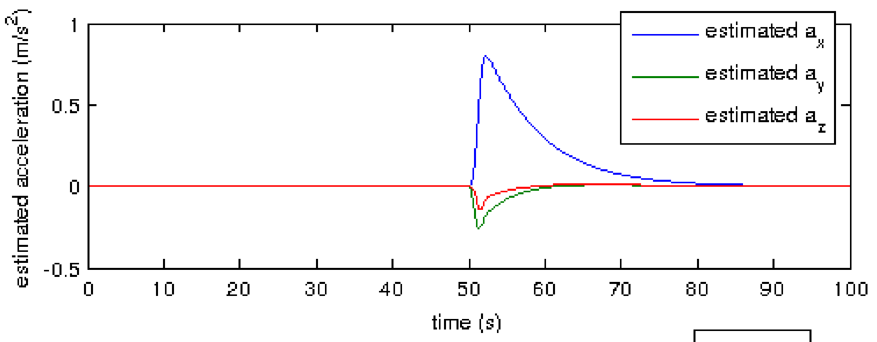


FIGURE 8.3: Simulink OKAN angular velocities.

- the subject is not linearly accelerated in any way, but, near 50s, he feels to be accelerated due to the post-rotational tilt effects. In particular small negative accelerations are perceived along the y and z axes, while a bigger positive longitudinal acceleration is felt



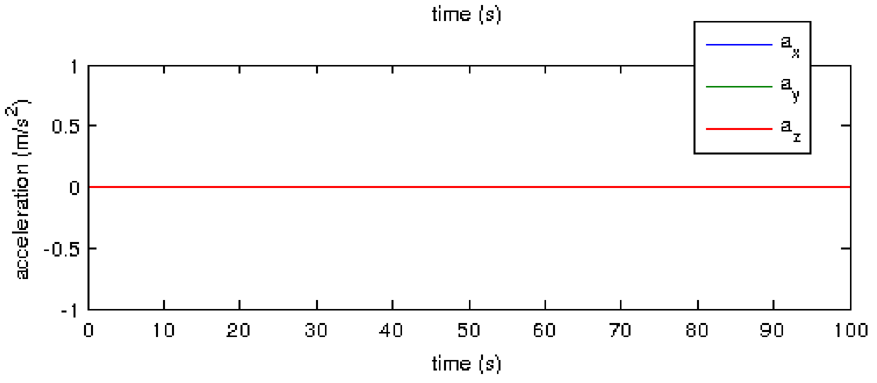
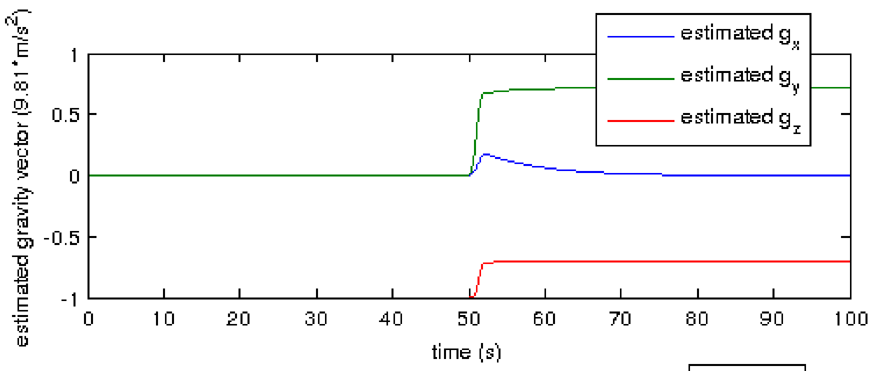


FIGURE 8.4: Simulink OKAN accelerations.



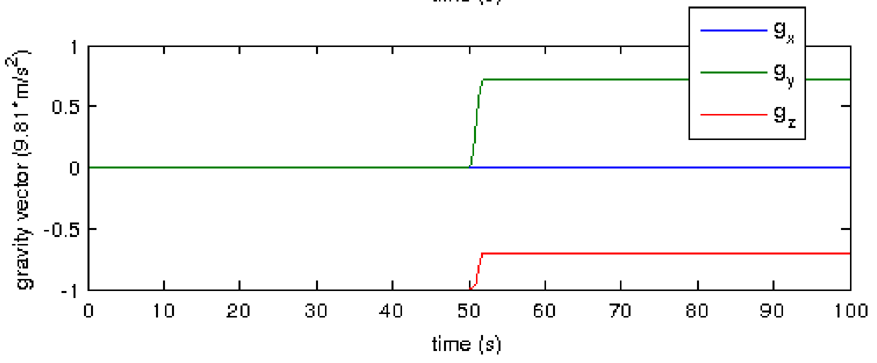


FIGURE 8.5: Simulink OKAN gravity vector.

- the estimated gravity direction vector does not replicate what the real gravity vector does. Here, like in the angular velocities case, there is a feeling about a pitch rotation, that is a post-rotational tilt effect too and, in fact, this false sensation slowly decays with the time constant of the system.

Looking at these figures, it is possible to clearly recognize the OKAN and its consequences. This behaviour can be explained by the physical structure of the vestibular

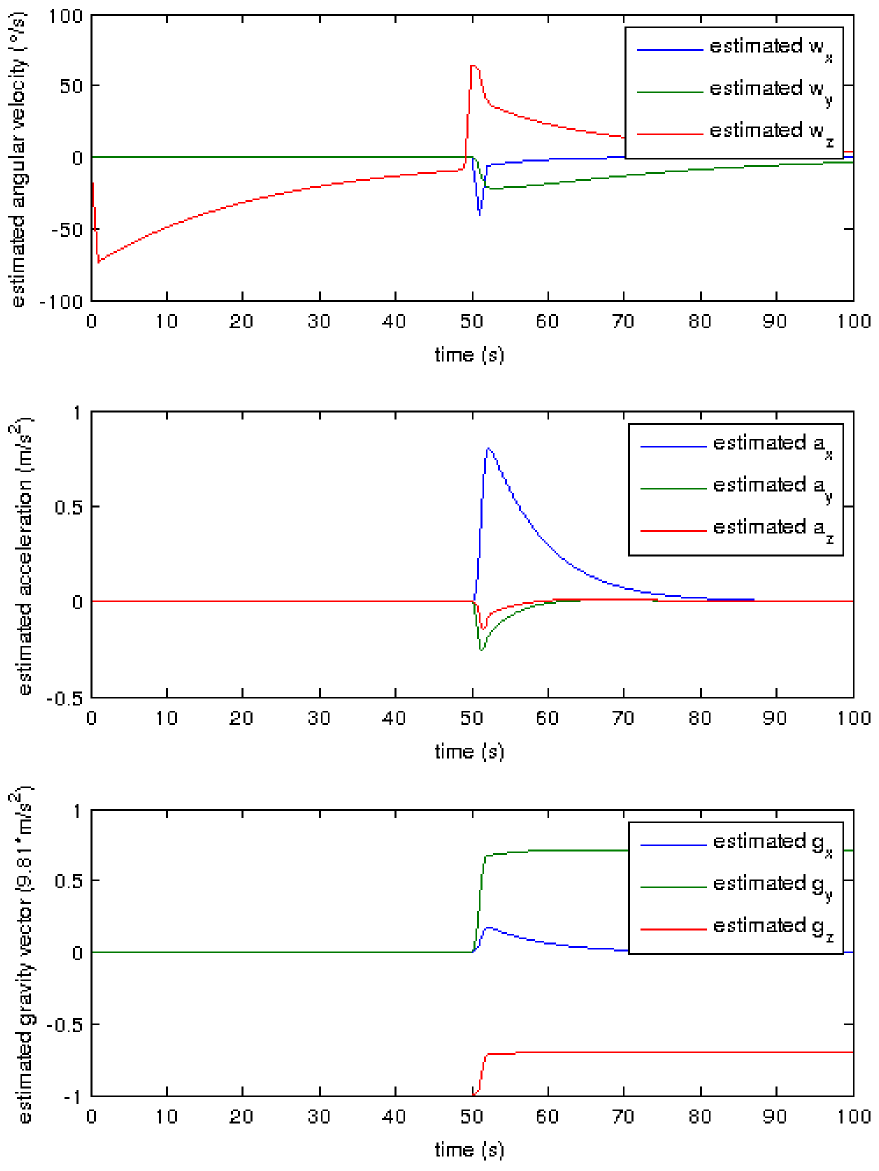


FIGURE 8.6: ACADO OKAN test results.

organs, by the SCCs in particular, as said in Chapter 4.

ACADO results The final model has also visual cues that are not used in the sensory conflict model by Merfeld, so, to satisfy the darkness hypothesis, the visual gains $k_g v$ and $k_{\omega v}$ are set to zero during these tests. Then, it is possible to replicate the same routine executed with the Simulink model by using the model implemented with the `ACADO toolkit`.

Figure 8.6 shows the results of this simulation. It is clear that qualitatively the results presented in Figures 8.3, 8.4, 8.5 are perfectly replicated, but also quantitatively they are exactly the same as verified numerically by simply subtracting the two different signals and obtaining a null vector.

8.1.2 Off-vertical axis rotation

Off-Vertical Axis Rotation (OVAR), in darkness conditions, induces, at small tilts of the rotation axis (5 to 45 degrees), continuous horizontal nystagmus in humans, [35]. The horizontal slow eye velocity has two components: a mean velocity in the opposite direction of head rotation and a sinusoidal modulation around the mean.

The subject is put into darkness, he is first rotated by a roll angular velocity till reaching an inclination of about 45° and then a constant yaw velocity of $100°/s$ is applied.

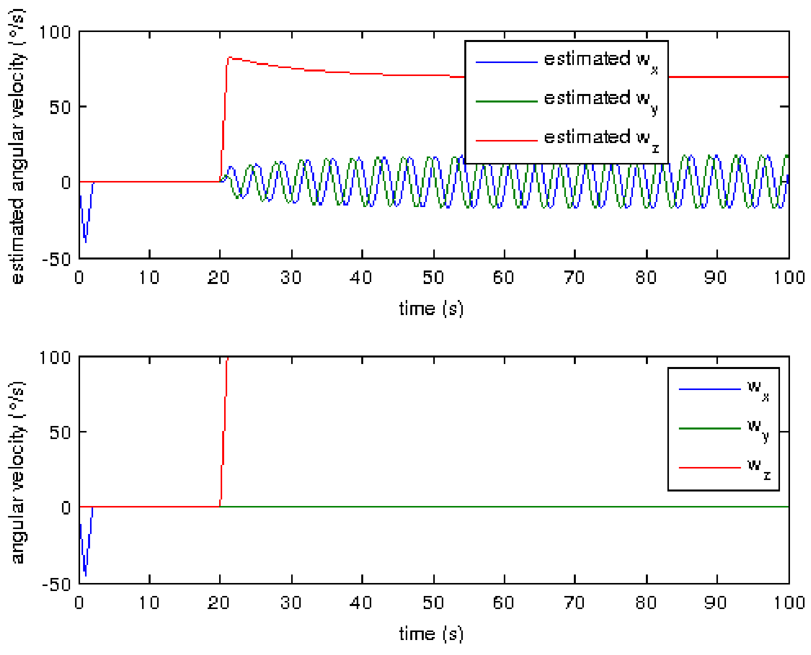


FIGURE 8.7: Simulink OVAR angular velocities.

Simulink results In Figures 8.7, 8.8, 8.9, it is possible to see the original, Simulink implemented, Merfeld's model results of the OVAR test just introduced. In these figures some comparisons between real variables and the estimated ones are represented. The effects of this test on the perception system are summarized:

- as during the post-rotational tilt, fast angular velocity changes are well perceived whereas the constant value of yaw rotation decays slowly but it seems to stabilize about 70 degrees per second. In fact, thanks to the initial roll rotation, the subject feels some sinusoidal roll and pitch angular velocities together with of the yaw rotation and this phenomenon makes the perceived yaw rotation to stop decaying
- linear accelerations are heavily influenced by this test. Sinusoidal longitudinal and lateral accelerations are perceived while the actual acceleration is always null. In fact, the yaw rotation is around an axis that is no more parallel to the real gravity vector and this causes a bad estimation of the gravity by the driver that influences the perceived linear acceleration
- the estimated gravity vector orientation, in this case, is qualitatively well estimated. But if it is analysed with more accuracy, it reveals that it has small magnitude deviations against the actual gravity vector and also a small delay. These facts are responsible for the wrong estimation of the linear acceleration stated above.

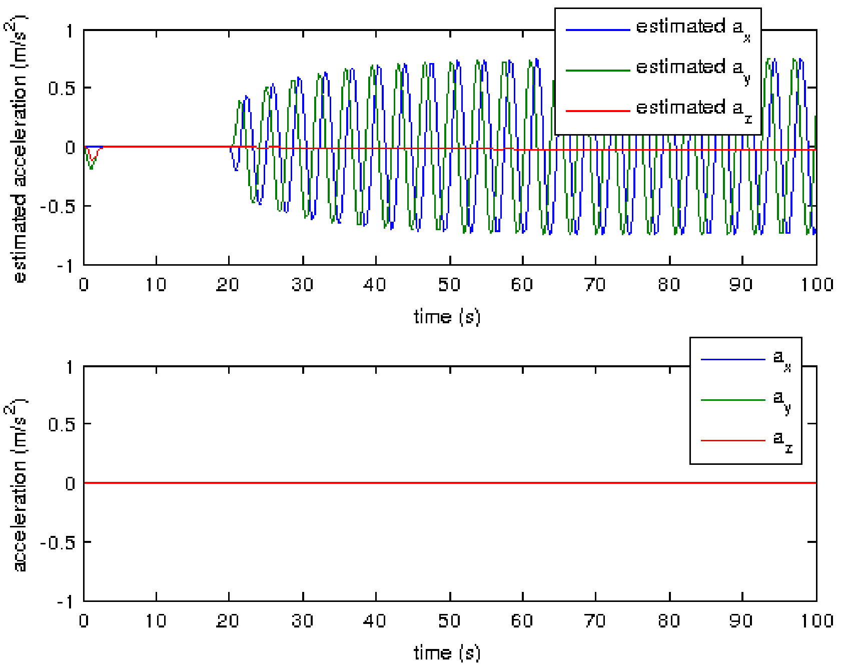


FIGURE 8.8: Simulink OVAR accelerations.

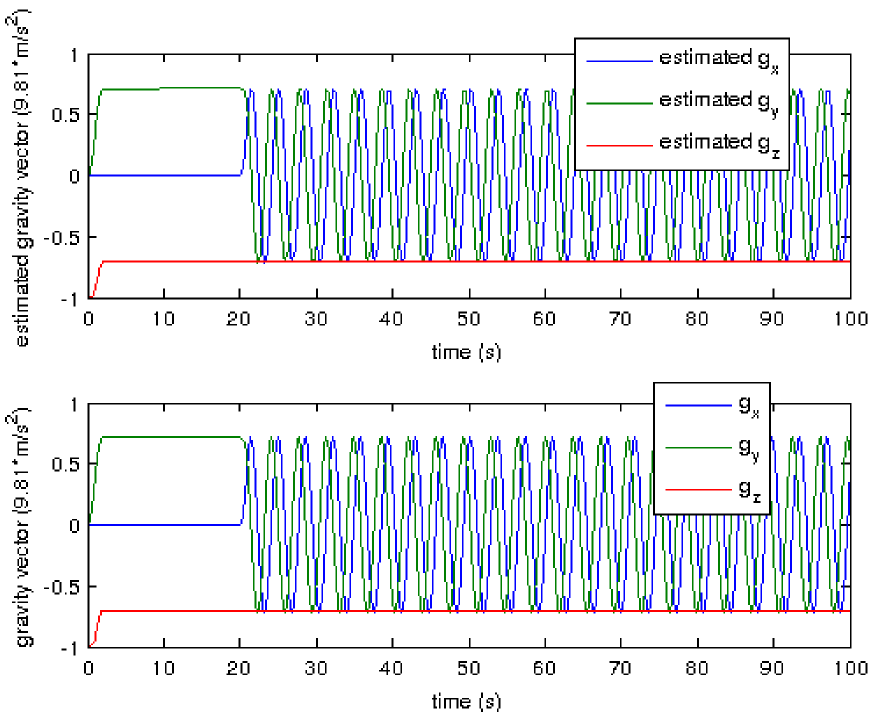


FIGURE 8.9: Simulink OVAR gravity vector.

ACADO results Also in this case, to satisfy the darkness hypothesis, the visual gains k_gv and $k_{\omega v}$ are set to zero during this test. Then the same routine of the Simulink OVAR test is replicated in the `ACADO` model.

Figure 8.10, show this simulation results. It is clear that the results presented in Figure 8.7, 8.8, 8.9, are perfectly replicated like the ones of the post-rotational tilt test.

This last test finally demonstrates that the final model used in this thesis and implemented with the `ACADO toolkit` can replicate the Merfeld's sensory conflict model dynamics even if it uses a different implementation approach. However, it is interesting to compare some computation time performances of the Simulink and of the ACADO model implementations.

Test	Simulation time (s)
Simulink post-rotational tilt	1.4958
ACADO post-rotational tilt	0.0833
Simulink OVAR	1.7347
ACADO OVAR	0.0881

TABLE 8.1: Mean simulation time during the different tests.

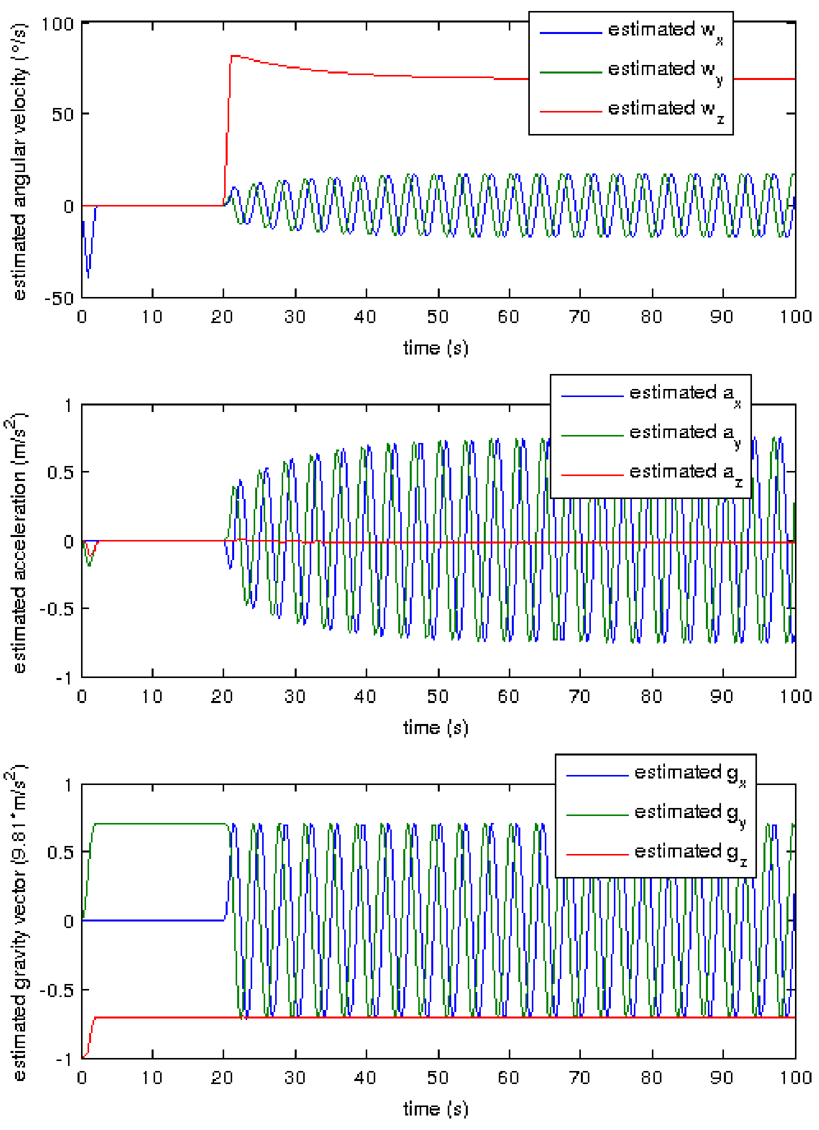


FIGURE 8.10: ACADO OVAR test results.

From Table 8.1, it is clear that, in the same conditions, the `ACADO` approach is at least 10× faster than Simulink one. This result has great importance in this work, and it is going to be even more important during control tests.

8.2 Control

In this section some tests about the automatic control are presented, i.e. the NMPC is set up and the control inputs are computed directly by the controller (Optimal Control

Problem (OCP) solver) with a control/prediction horizon of twenty steps (0.2s) at 100Hz. The purpose is to track the references for the system provided in the cost function defined in Section 7.4.1, and to find the best set of weighting values to obtain the optimal workspace exploitation.

These tests are done with the *complete* model, including the visual cues, that were disabled during the simulations made in the previous section.

8.2.1 Tilt coordination

As briefly explained in Chapter 2, the tilt coordination has an important role in Motion Cueing Algorithms because it allows to track some references that, using only platform longitudinal translations, would be impossible to reproduce. So it is crucial for the NMPC to achieve a good tilt coordination mechanism.

Regarding this aspect, the final model is deeply different from previous works. Usually, tilt coordination has to be mathematically implemented into the model, namely the system needs to know that it can use also the tilt coordination to track the references. Instead, with this non-linear model, this procedure is no more necessary. In fact, the system itself has implicit knowledge of the influence of the platform rotations on the perceived linear acceleration and angular velocity, so it applies the tilt coordination in a totally automatic way.

For example, in Figures 8.11, 8.12, 8.13, a constant longitudinal linear acceleration reference is given to the NMPC. The tracking of a constant longitudinal acceleration in a limited-space driving simulator is clearly an impossible task. However, it is interesting to analyse how the controller behaves trying to solve the OCP and how it exploits the tilt coordination:

- the platform clearly accelerate longitudinally but its acceleration magnitude must be as small as possible because the simulator has a limited workspace. In fact, first the platform accelerate rapidly, then the magnitude of the acceleration decreases towards zero. During this movement the estimated acceleration tracks almost perfectly its reference

- to compensate the limited acceleration of the platform, tilt coordination is performed. A negative pitch angle is applied to the platform and this makes the driver feel an augmented longitudinal acceleration (as explained in Section 4.2) for the effect of gravity, so the constant reference is tracked very well even if the platform acceleration is low than the requested one

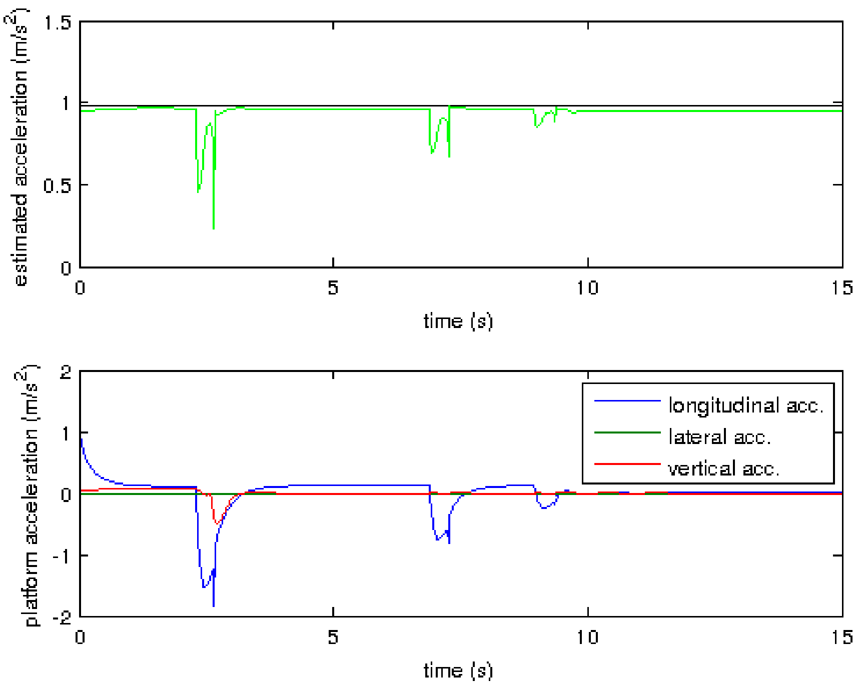


FIGURE 8.11: Estimated acceleration and platform acceleration in the tilt coordination test. The black line represents the reference, the green one the output.

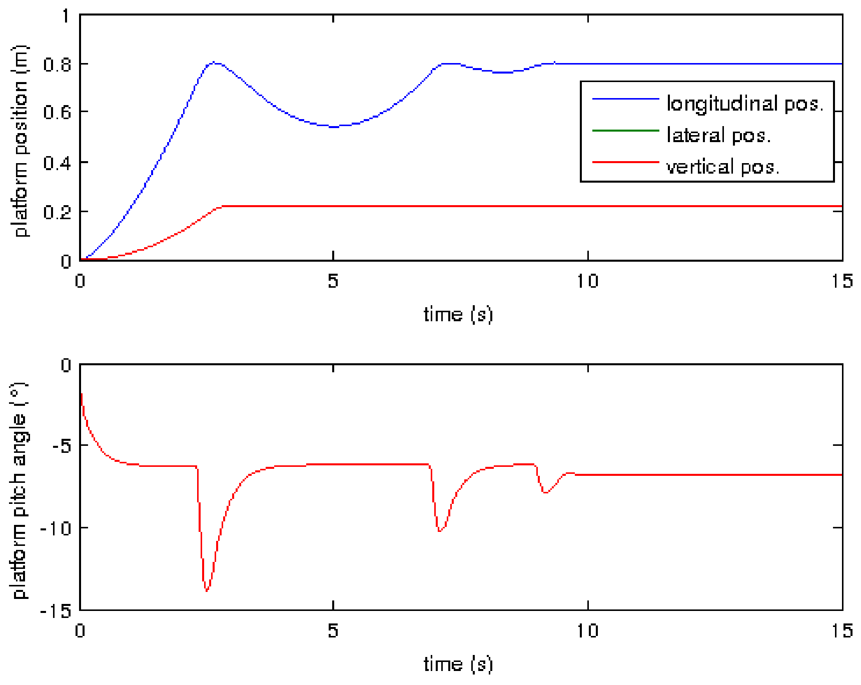


FIGURE 8.12: Platform position and pitch angle in the tilt coordination test.

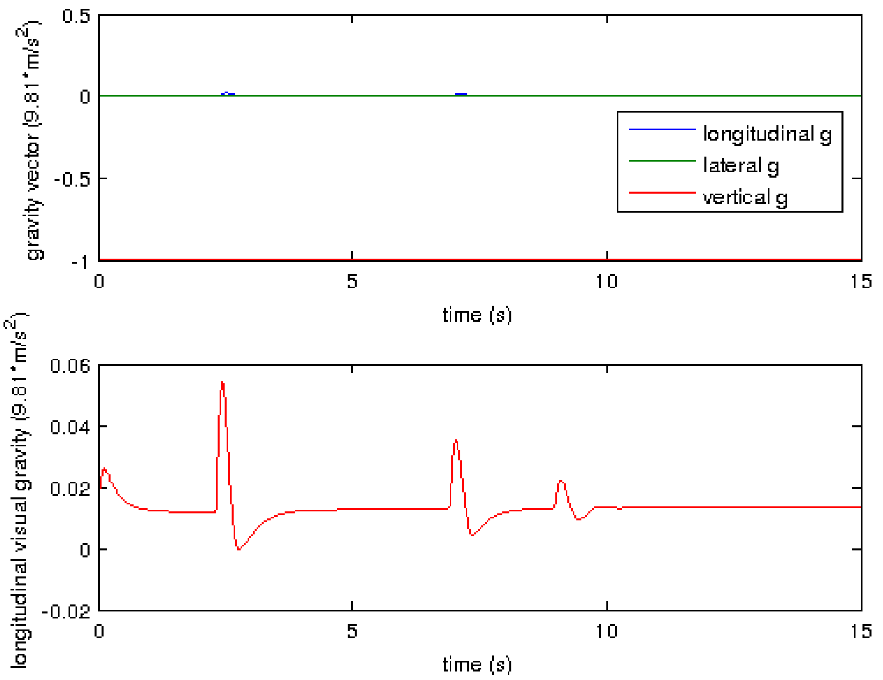


FIGURE 8.13: Estimated gravity vector and longitudinal visual gravity in the tilt coordination test.

- when the platform reaches its edge, a negative acceleration must take place to decelerate and then to stop. At this point the platform can only play with the angular velocity to obtain a pitch angle that has to give the right contribution to track the reference
- during all these movements the perception of gravity is almost always vertical, thank to the visual cues. In fact, the negative pitch angle of the platform is compensated by the graphics that simulates an opposite (positive) pitch rotation, so the driver does not feel tilted.

The simulation time for this test is about 25s to compute 15 seconds of control. Clearly it is not real-time but the performances are influenced by the "impossible" task requested to the controller, so it is not unattended that the computation time grows up.

8.3 Real circuit lap

In this section a more realistic situation is analysed. From real data acquired during a circuit lap, the references are generated and the NMPC is tested to understand how such a control strategy based on the final model behaves in a real driving simulator session, see Chapter 7.

8.3.1 References generation

First of all, it must be clear that the references given to the NMPC are defined in the driver coordinate frame, that is consistent with the acquired data of the circuit lap because all the sensors were fixed to the car chassis. But the requested references are not the real angular velocity, linear acceleration and gravity vector, but their respective estimated values. So, before starting the simulation, the circuit lap data must run as an input to the model. This process produces the outputs, namely the estimated linear acceleration, the estimated angular velocity and the estimated gravity vector that are the right references the NMPC has to track.

The data does not have visual cues, so, during the references generation process, the visual angular velocity control of the model is always fixed to zero. This means that the graphics follows the cockpit movements and gives no more information to the driver. See Figures 8.14, 8.15, 8.16, to analyse the obtained results.

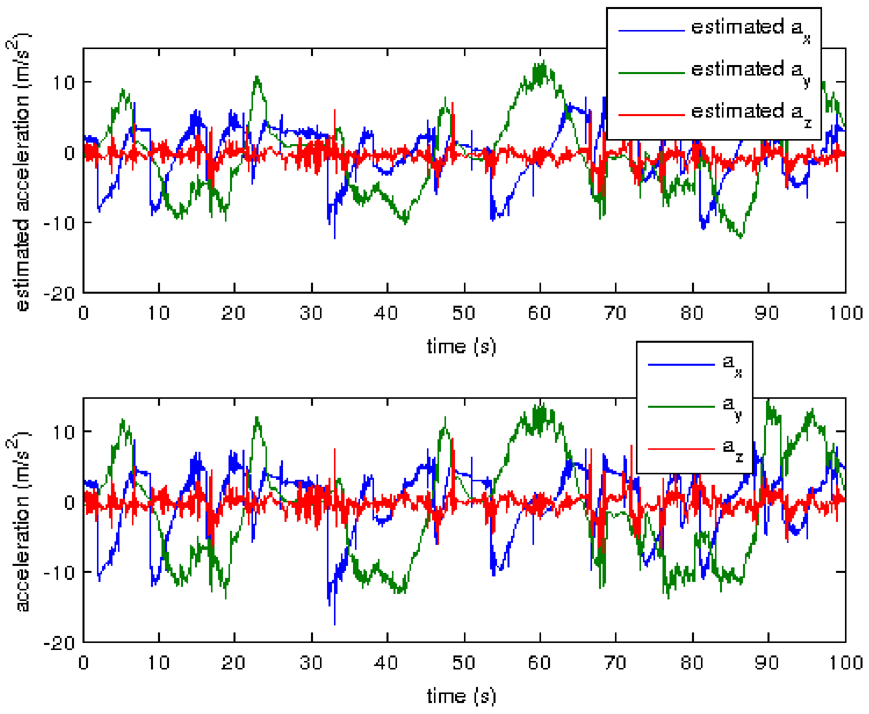


FIGURE 8.14: Linear acceleration reference generation.

A fast comparison between the actual and the perceived references makes clear that there are a lot of differences between them. This is not unexpected, in fact, in Section

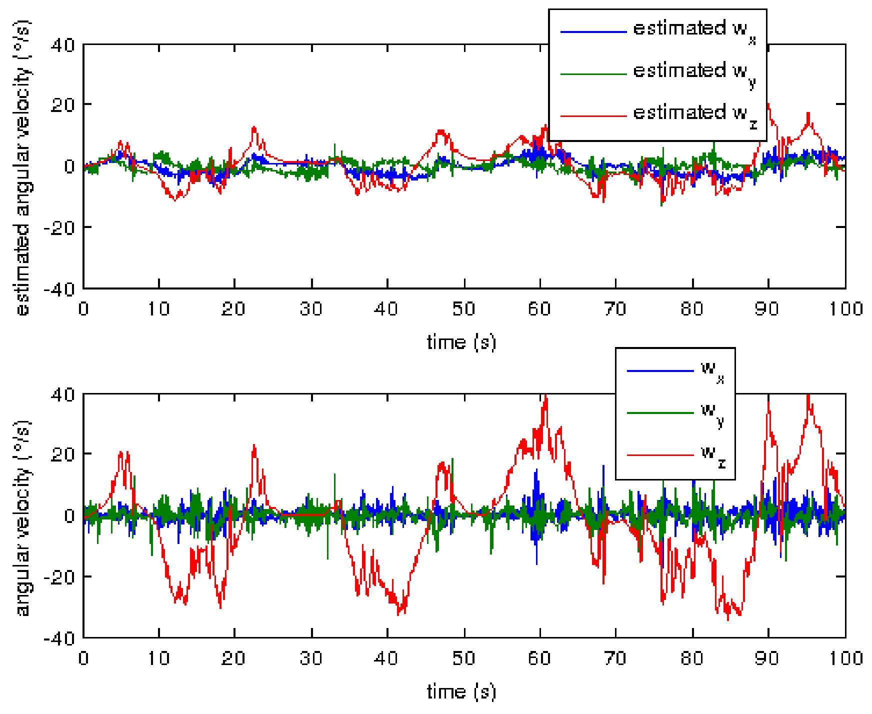


FIGURE 8.15: Angular velocity reference generation.

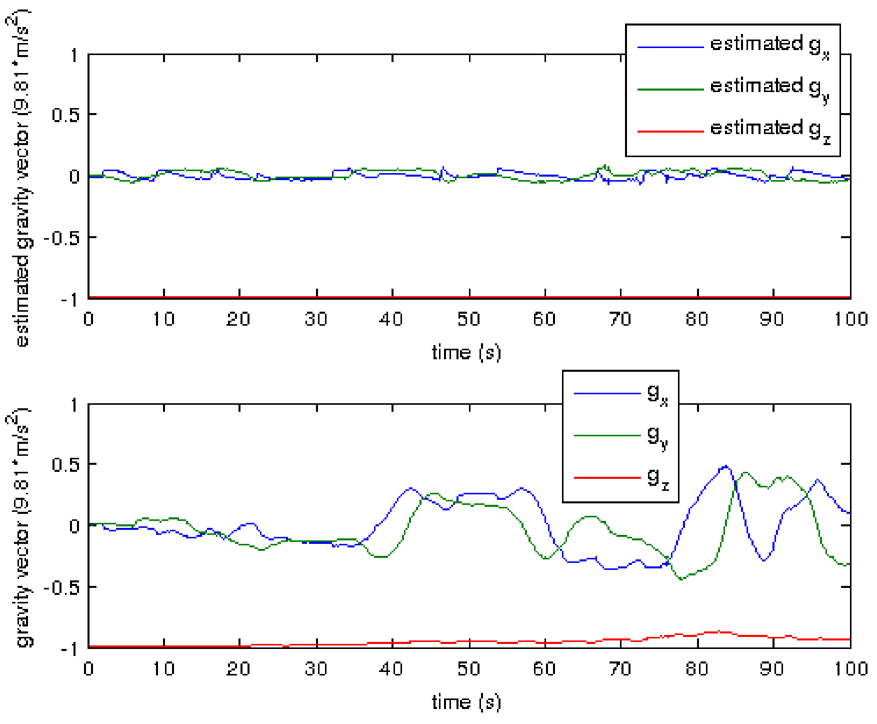


FIGURE 8.16: Gravity vector reference generation.

8.1, it has been illustrated that actual and estimated values of the same variable can be quite different. Some considerations have to be done:

- estimated linear acceleration references is very similar to its respective real values. The only noticeable thing is a bit smaller magnitude along the whole the lap
- estimated angular velocity references are much different from the real ones. The trend is clearly the same, but the magnitude is greatly reduced thank to the imposed condition on the visual angular velocity control, that, making the graphics follow the driver, makes him feel smaller rotations
- similar consideration can be done about the estimated gravity vector. In fact, it remains almost always equal to the initial vertical position, that means the driver always feels to stand still, a clear consequence of the imposed visual cues.

Some interesting considerations can also be done if the model is taken into consideration without the visual cues, such as the pilot would drive into the darkness and have only the vestibular system information to understand what is happening.

In Figure 8.17, it is possible to point out that the references generated “into the darkness” are quite different from the ones with visual cues, Figures 8.14, 8.15, 8.16:

- linear accelerations are a bit sharper, but the similarity to the respective reference with the visual cues is evident
- angular velocities are quite different: they oscillate more and their magnitude is almost twice of the respective reference with the visual cues. All these behaviours are consequences of the absence of graphics, in fact a static graphics reduces the feeling of rotation whereas its absence, like being into the darkness, produces a spatial disorientation feeling
- gravity vector reference is completely different w.r.t. the previous case, the driver feels to rotate continuously, which does not happen with enabled visual cues, situation when the pilot feels almost always in vertical position.

After these interesting considerations, it is possible to go on with the real circuit lap simulation. The next step of the process is the reference scaling. The purpose is to find a set of parameters that, together with the weights of the OCP, allow to fairly track the references. These scaling factors are chosen to replicate the results of previous works, in this way it is simply to make a comparison between different approaches (see Chapter 9). Used values are reported in Table 8.2.

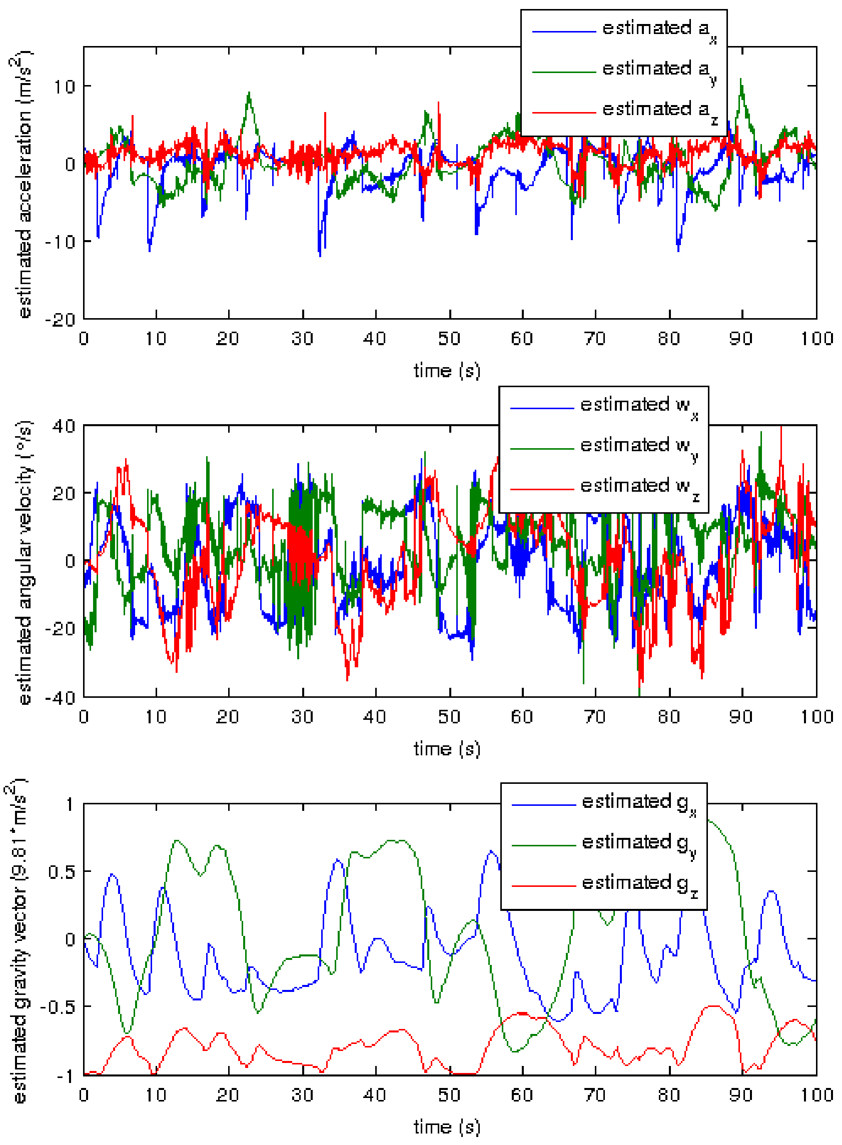


FIGURE 8.17: References generation into darkness, without visual cues.

Variable	Scaling factor
linear longitudinal acceleration	0.05
linear lateral acceleration	0.06
linear vertical acceleration	0.025
angular roll velocity	0.85
angular pitch velocity	0.75
angular yaw velocity	0.9

TABLE 8.2: Scaling factors for the NMPC references.

8.3.2 NMPC

Now the final references are available and it is possible to initialize the NMPC. The values of the weighting matrices must be adjusted until the results are good enough. Weighting values are different for each application and for each simulation. These values are very important to understand how the model works and, for this reason, they are going to be analysed later on.

Fixed graphics The first performed test is a NMPC without the graphics control. In other words, it is used a fixed graphics in the vertical position of the driver coordinate frame, such as the pilot always has a sensation to stand still, as explained in the references generation section. The aim of this hypothesis is to test the model with no information given by the visual cues while trying to track the references with an appropriate accuracy.

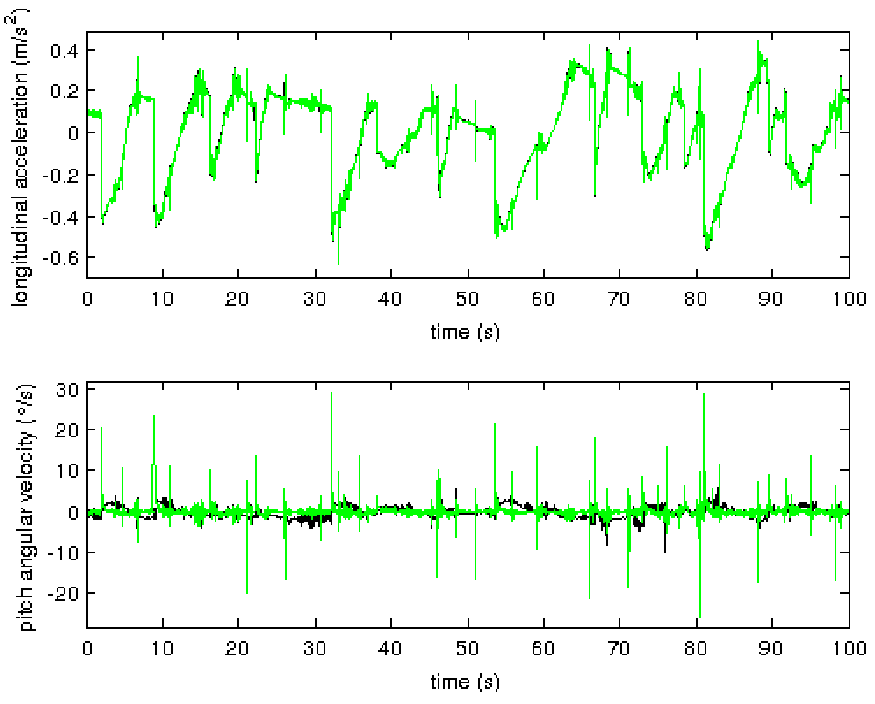


FIGURE 8.18: NMPC perceived longitudinal acceleration and pitch angular velocity output. The black line is the reference and the green line is the output.

In Figures 8.18, 8.19, 8.20, 8.21, it is possible to see the results of this simulation. Some comments about them:

- the tracking of the accelerations is quite well achieved, it is almost perfect for the longitudinal and lateral accelerations

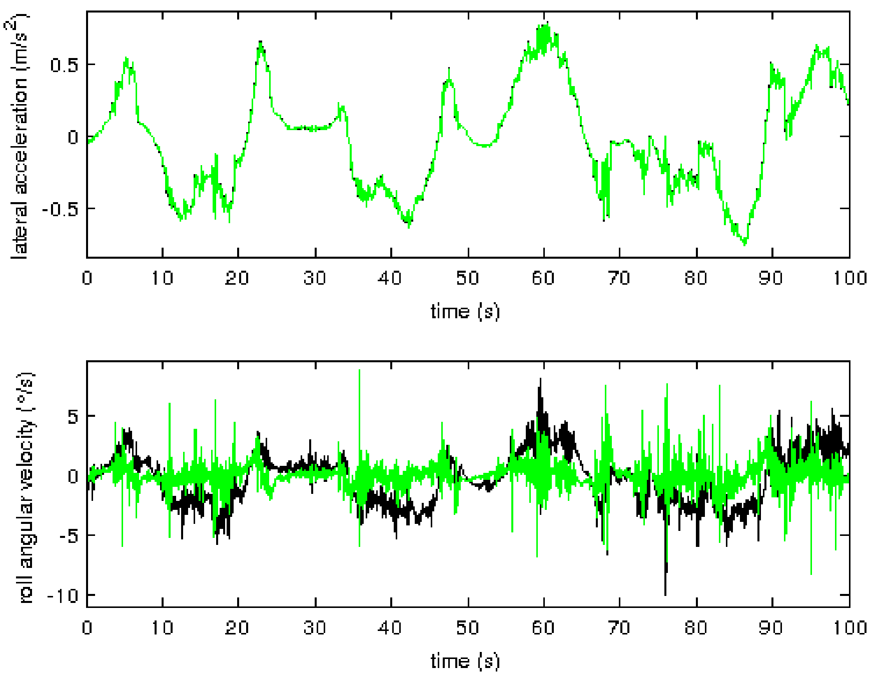


FIGURE 8.19: NMPC perceived lateral acceleration and roll angular velocity output. The black line is the reference and the green line is the output.

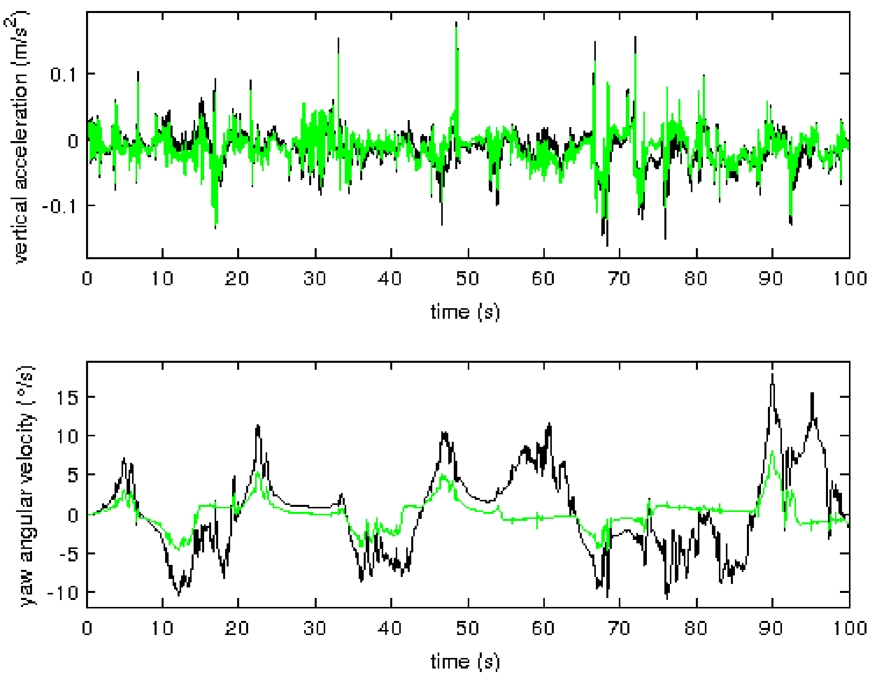


FIGURE 8.20: NMPC perceived vertical acceleration and yaw angular velocity output. The black line is the reference and the green line is the output.

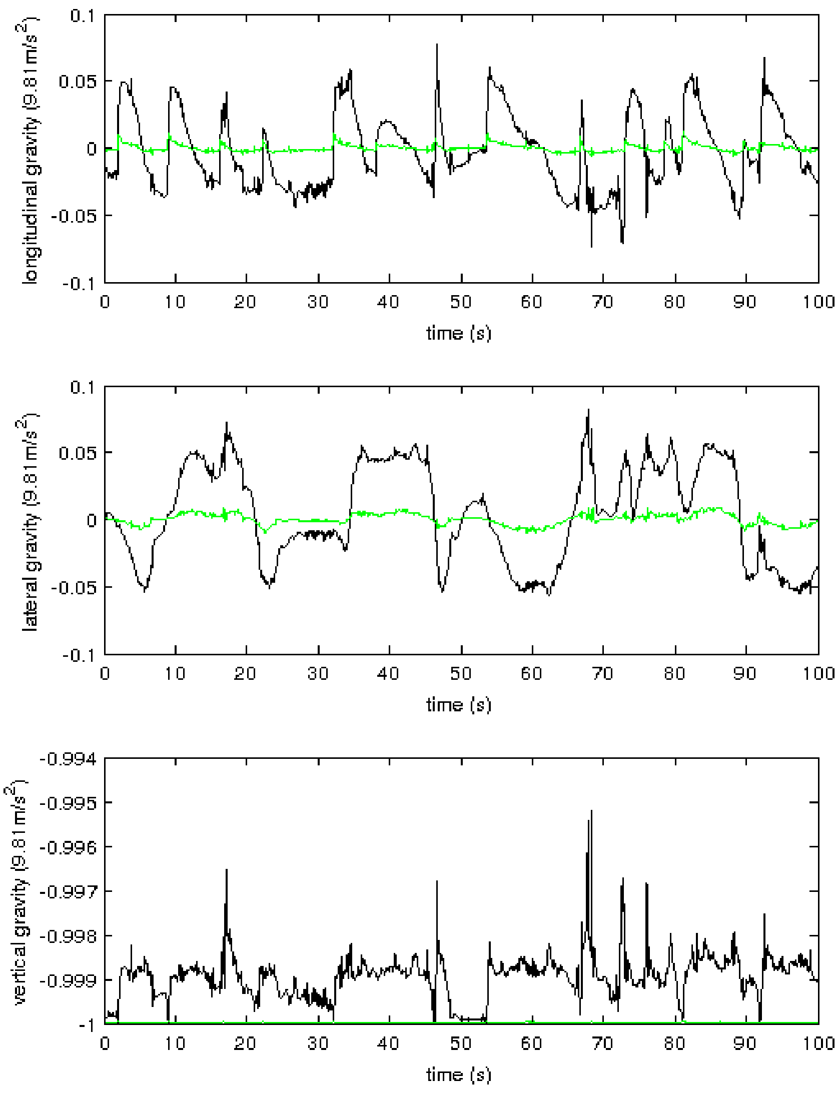


FIGURE 8.21: NMPC perceived gravity vector output. The black line is the reference and the green line is the output.

- there are some fast changes in the pitch and roll angular velocities that could create some problems. In general, the tracking of the angular velocities is worse than the linear accelerations tracking
- the fixed graphics allows the driver to obtain an almost vertical feeling of gravity, and that is a good thing. In fact, even if the gravity reference is not always vertical, the estimated output has only small direction displacements that can be assumed to be under the perception threshold (this point will be discussed later, in Section

8.3.3), so the driver always feels to stand still, i.e. the driver does not perceive that the platform is tilting.

As a last observation, the computation time is taken into account. The simulation lasts about 180 seconds for a a hundred seconds of control, so the algorithm is running at 1.8×. Again it is not real-time, but in the next section it is going to show that if the graphics is enabled (three more DOFs) the control is easier and so the computation time is lower.

Graphics enabled Now, the graphics control is enabled, so there are three more DOFs (rotations of the graphics, namely the visual angular velocity control $\vec{\omega}_v$). With only this change, the simulation is repeated and it is analysed how this improvement influences the control strategy and the tracking of the references.

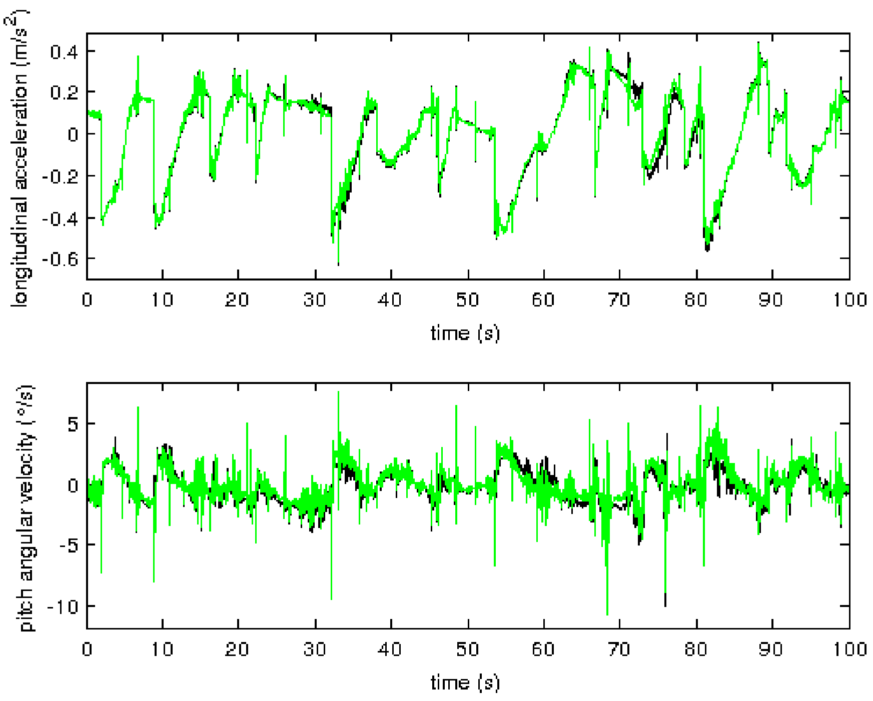


FIGURE 8.22: NMPC perceived longitudinal acceleration and pitch angular velocity output. The black line is the reference and the green line is the output.

Looking at Figures 8.22, 8.23, 8.24, 8.25, it is possible to resume some considerations:

- the tracking of the linear accelerations is, as before, well achieved. There are no significant peaks and the error between the output and the respective reference is almost always near zero

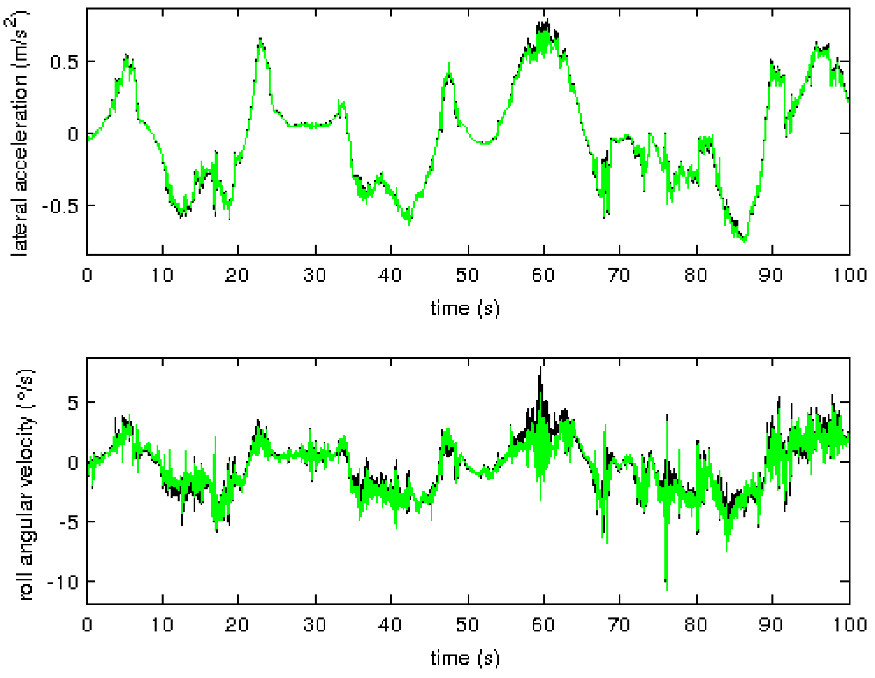


FIGURE 8.23: NMPC perceived lateral acceleration and roll angular velocity output. The black line is the reference and the green line is the output.

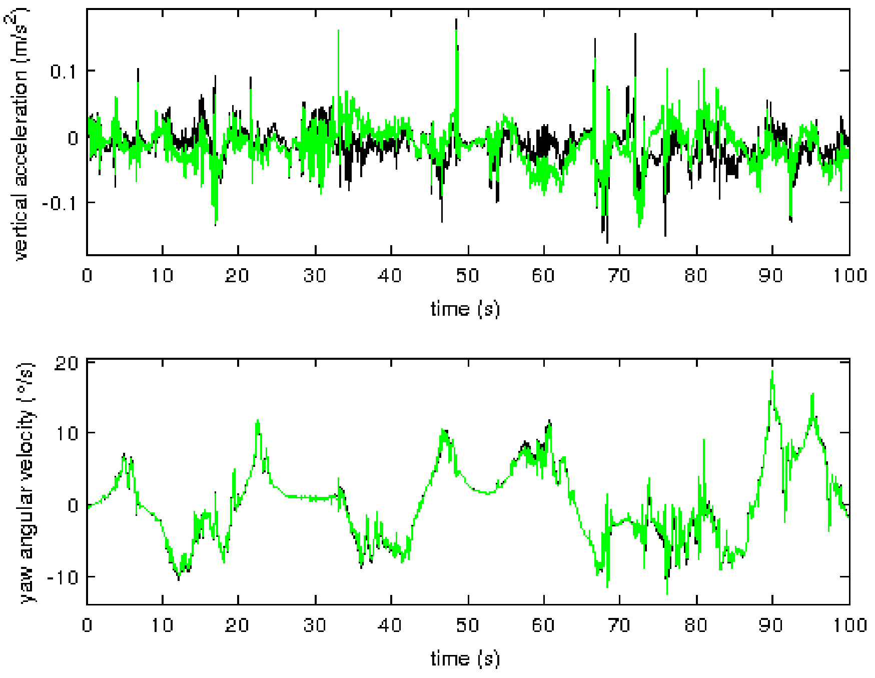


FIGURE 8.24: NMPC perceived vertical acceleration and yaw angular velocity output. The black line is the reference and the green line is the output.

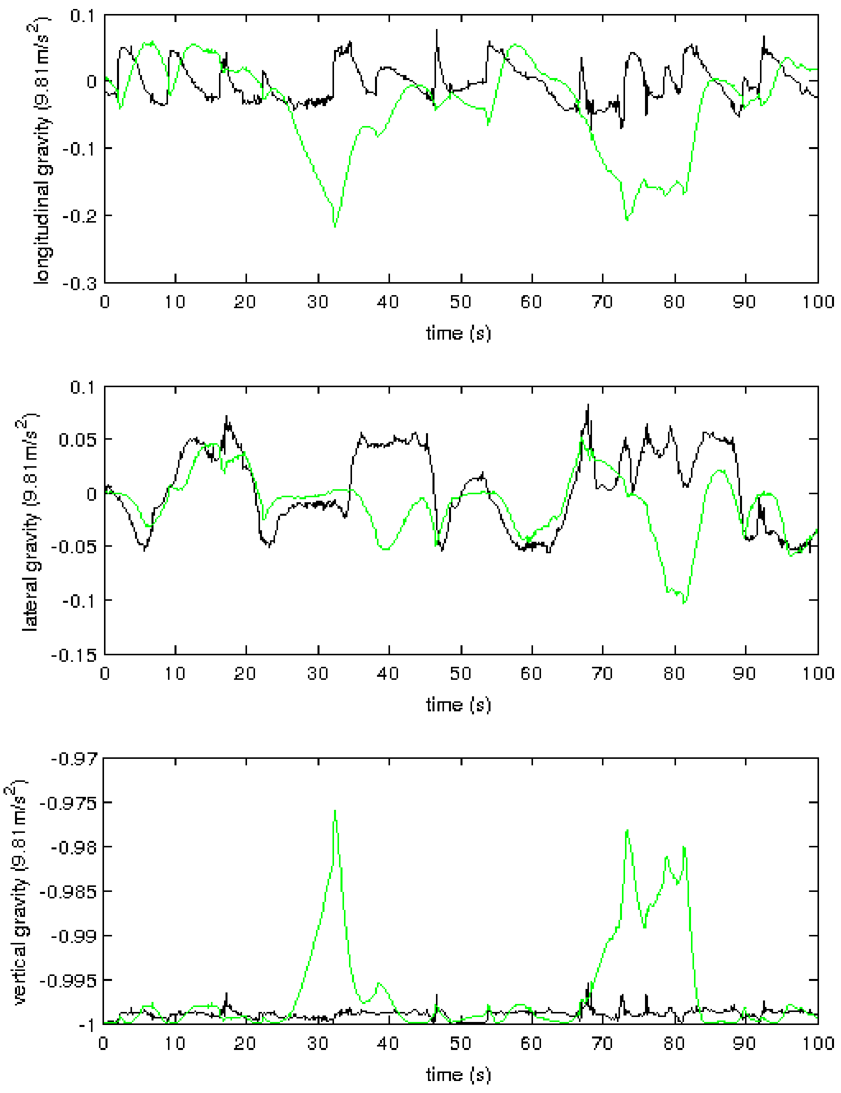


FIGURE 8.25: NMPC perceived gravity vector output. The black line is the reference and the green line is the output.

- angular velocities are tracked very well with the graphics enabled, that is an important improvement from the previous simulation
- the gravity feeling is at least acceptable, the gravity direction is less constant than before because the graphics rotates too. It has some differences from its reference, but it always stays under the 0.2g value. This value can have an important role, that is if it stays under the perception threshold or not; it will be discussed later in Section 8.3.3.

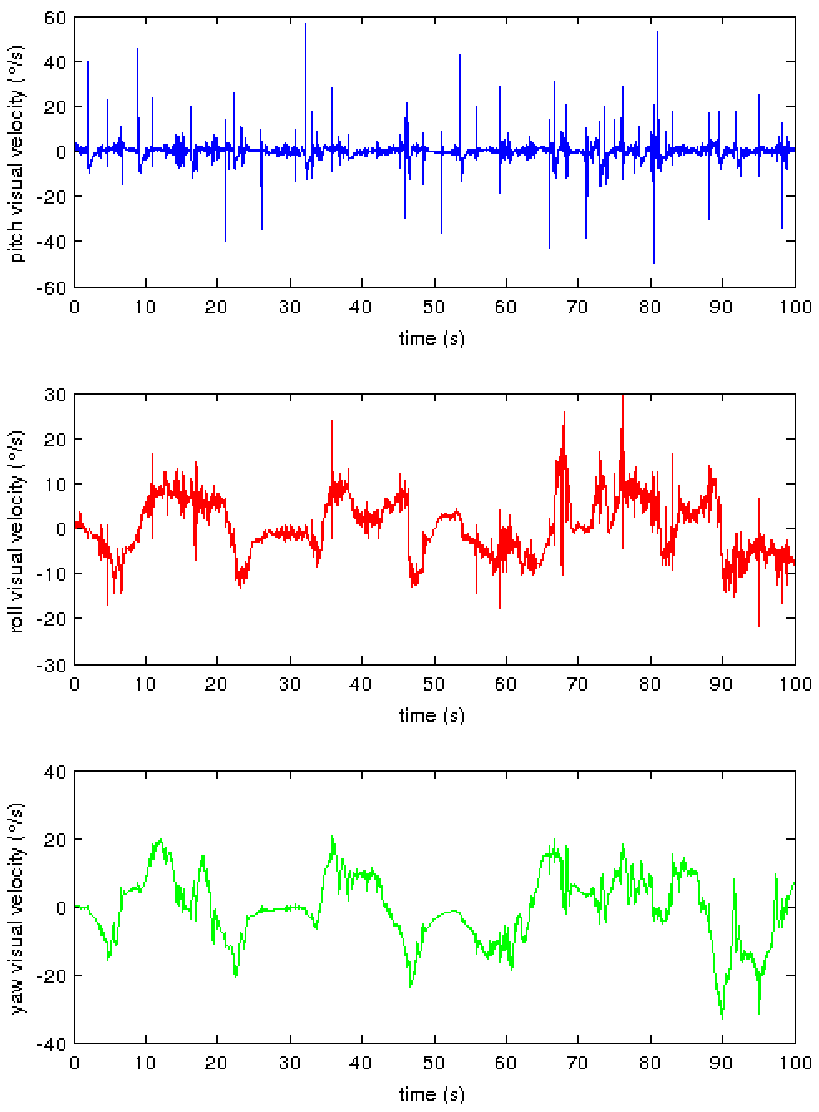


FIGURE 8.26: Visual angular velocity control ω_v.

Looking at the visual control ω_v (see Figure 8.26), it is possible to comprehend how it works together with other inputs to achieve the references tracking.

The visual angular velocity behaviour tries to compensate for the perceived angular velocities, in fact, if these two trends are compared, it is clear that one follows the opposite of the other. This evidence demonstrates their clear correlation. This interesting fact is represented in Figure 8.27, where perceived angular velocities are overlapped by the opposite of the visual angular velocity control.

Looking at the computation cost of this last simulation, it is find out that, even if the problem is more complex (3 more DOFs), the simulation time is lower then before. In fact, to compute the control and simulate the system for a hundred seconds it takes about 120s, i.e. 1.2×. This is an encouraging result because the real-time goal is not so far and, if some improvements will be brought into the code, it is realistic that a real-time implementation could take place.

8.3.3 Considerations

Now it is possible to analyse the results obtained from these simulations. In particular, it is interesting to take a look at the weighting matrices, see Tables 8.4, 8.3, for each variable and to try to understand why a weighting value is bigger than another one.

Some considerations can be made:

- with both fixed or enabled graphics, weights of acceleration, velocity, position and controls about the z axis are larger than the other ones, that means it is more difficult to control and to track vertical acceleration. In fact the vertical workspace limit is smaller than the one of the others axes, and so it has to be weighted more, so the controller has to pay more attention on this dimension. This behaviour is even more accentuated with the graphics enabled, where these weights assume values even between ten to a thousand times bigger than values for the longitudinal and later movements
- weights for angular velocities and angles are smaller than the ones for linear movements. This can be explained by the different magnitude of these variables and by the bigger difficulty of tracking accelerations instead of the angular velocities. In fact, the limited translation workspace of the platform seems to be a tighter limit than its maximum angles. However, with the graphics enabled, more attention must be paid on the pitch and roll angular velocities tracking: their weights are five times bigger than the yaw velocity and than the angular velocities of the simulation with fixed graphics
- weights about the estimated gravity vector direction are, in both cases, set to zero because they seems not to heavily affect the tracking.

Gravity estimation takes an important role in these considerations. With fixed graphics the estimated gravity vector is almost always in vertical position (Figure 8.21), so the driver never feels rotated. With graphics enabled, the graphics is not in agreement with the driver coordinate frame, but in this way the estimated gravity vector has some

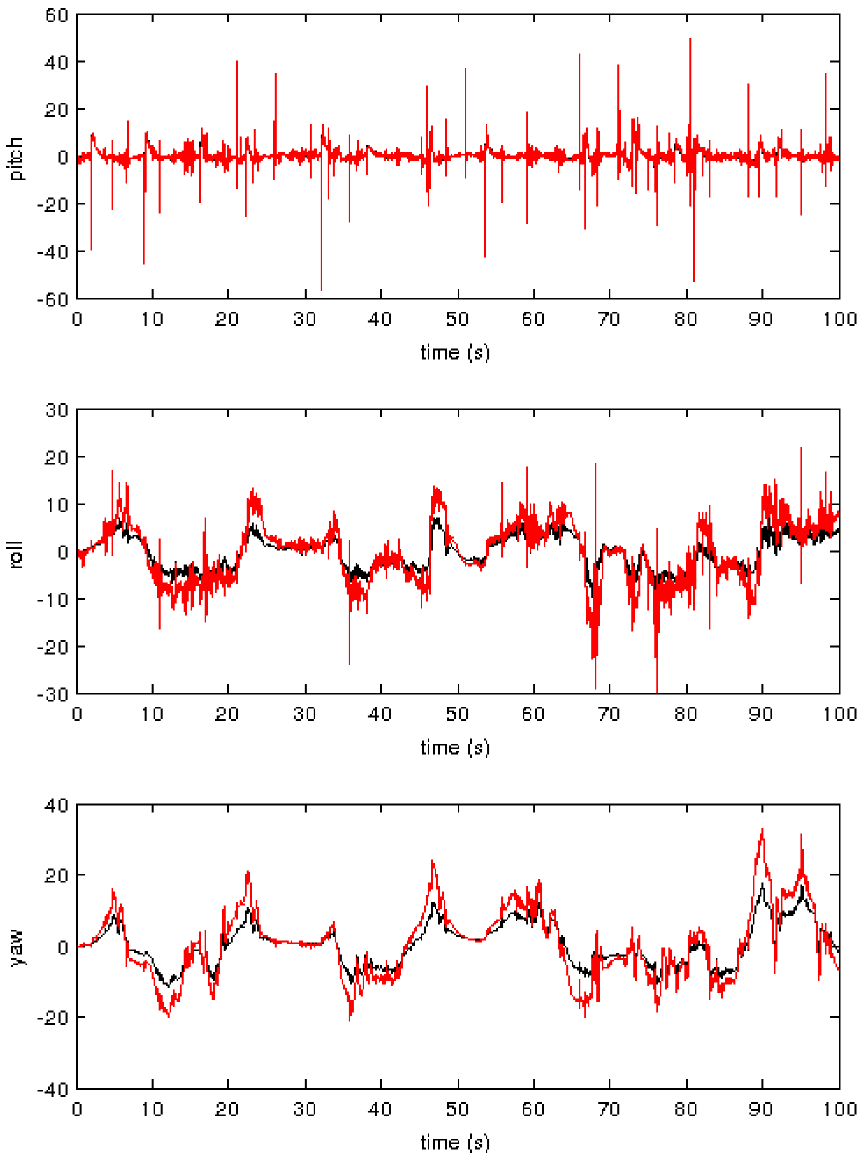


Figure 8.27: Estimated and visual angular velocities correlation. The perceived angular velocity graph (black line) is overlapped by the opposite of the visual angular velocity (red line).

oscillations that reach values up to $0.2g$ (Figure 8.25). It is not known if this value is small enough to stay under the perception threshold of the driver because there are no studies about this aspect. If heavier weight is put on $\overrightarrow{\hat{g}}$, the estimated gravity vector will better track its reference and will avoid too high values.

Variable	Weight value [x,y,z]
$\hat{\vec{\omega}}$	0.1, 0.1, 0.1
$\hat{\vec{a}}$	150, 200, 300
$\vec{e}_\omega$	0, 0, 0
$\vec{e}_a$	0, 0, 0
$\vec{e}_f$	0, 0, 0
$\vec{e}_{gv}$	0, 0, 0
$\vec{e}_{\omega v}$	0, 0, 0
$\vec{\theta}$	0.001, 0.001, 0.001
$\vec{p}$	50, 100, 200
$\vec{v}$	10, 50, 200
$\vec{\omega}$	0.01, 0.01, 0.01
$\vec{a}$	10, 10, 10
$\vec{g}_v$	0, 0, 0
$\vec{\omega}_v$	0, 0, 0
$\hat{\vec{g}}$	0, 0, 0

TABLE 8.3: Weighting values for the control with fixed graphics.

Variable	Weight value [x,y,z]
$\hat{\vec{\omega}}$	0.5, 0.5, 0.1
$\hat{\vec{a}}$	100, 100, 1000
$\vec{e}_\omega$	0, 0, 0
$\vec{e}_a$	0, 0, 0
$\vec{e}_f$	0, 0, 0
$\vec{e}_{gv}$	0, 0, 0
$\vec{e}_{\omega v}$	0, 0, 0
$\vec{\theta}$	0.0001, 0.0001, 0.0001
$\vec{p}$	50, 50, 5000
$\vec{v}$	10, 10, 1000
$\vec{\omega}$	0.1, 0.1, 0.1
$\vec{a}$	10, 10, 50
$\vec{g}_v$	0.1, 0.1
$\vec{\omega}_v$	0, 0, 0
$\hat{\vec{g}}$	0, 0, 0

TABLE 8.4: Weighting values for the control with graphics enabled.

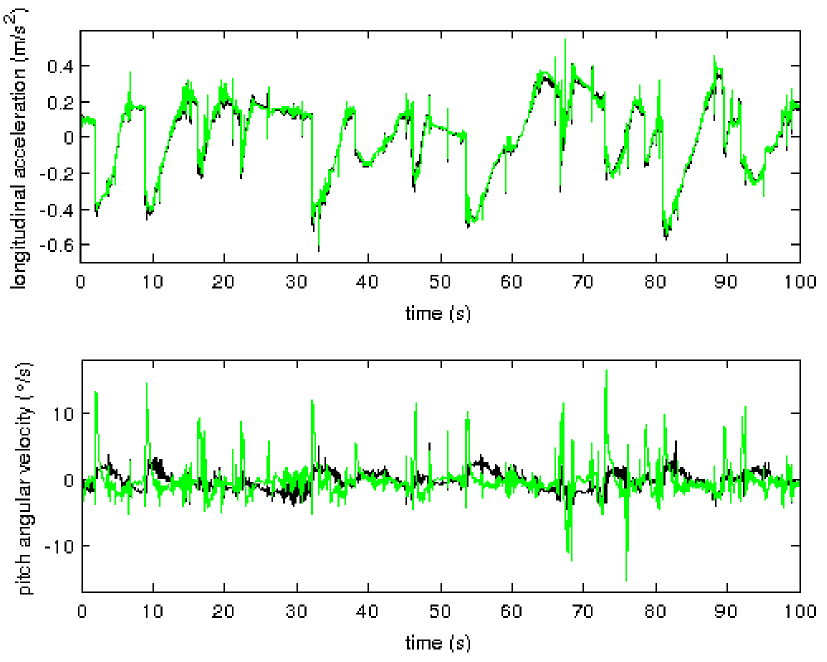


FIGURE 8.28: NMPC perceived longitudinal acceleration and pitch angular velocity output with low g displacement. The black line is the reference and the green line is the output.

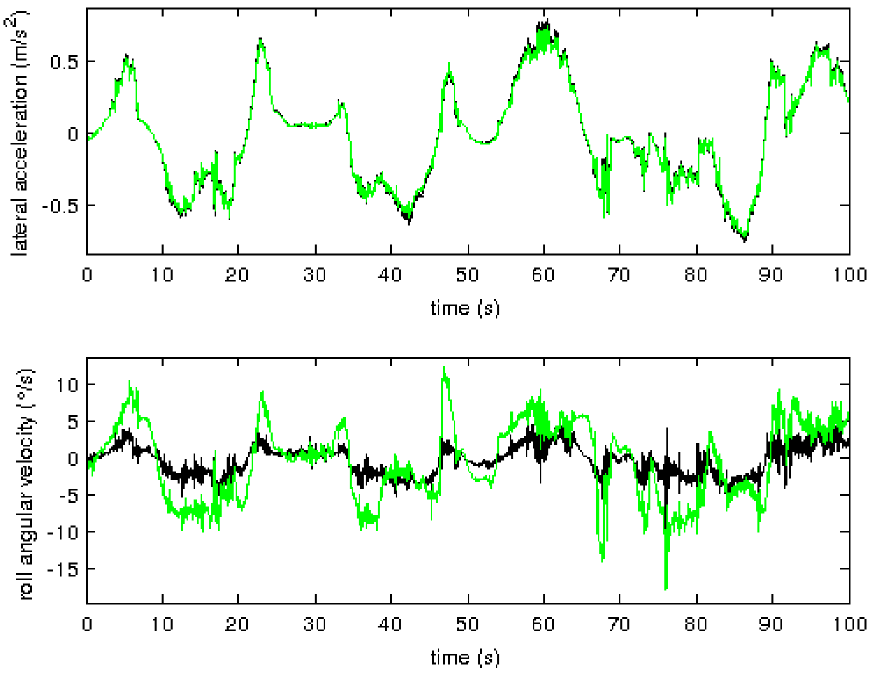


FIGURE 8.29: NMPC perceived lateral acceleration and roll angular velocity output with low g displacement. The black line is the reference and the green line is the output.

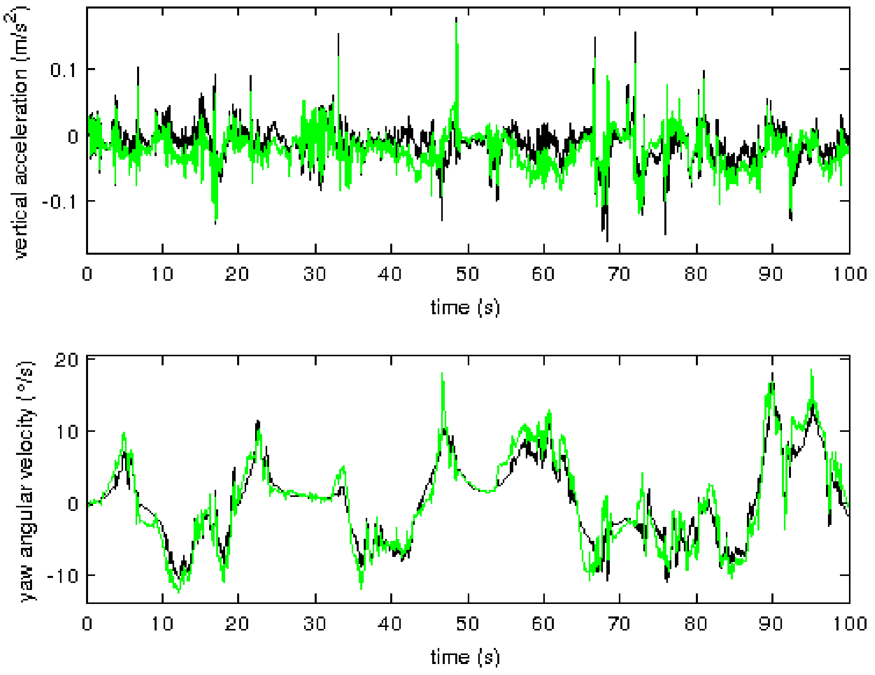


FIGURE 8.30: NMPC perceived vertical acceleration and yaw angular velocity output with low g displacement. The black line is the reference and the green line is the output.

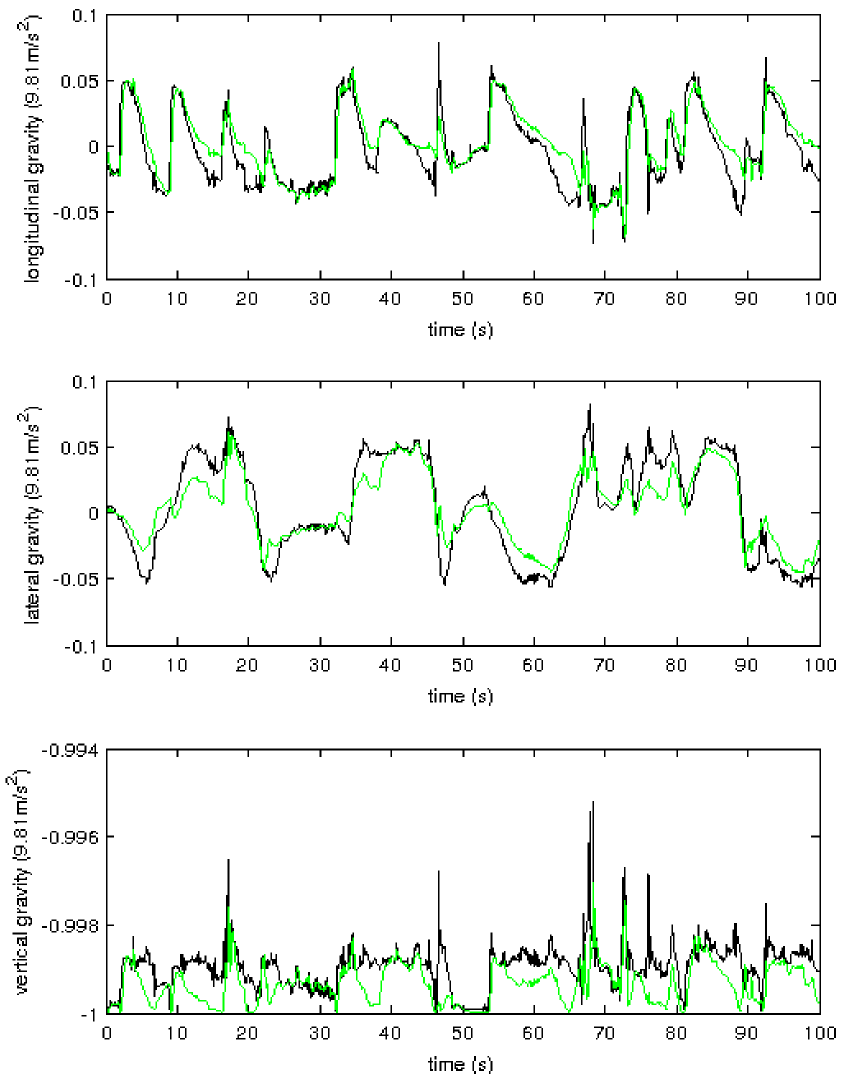


Figure 8.31: NMPC perceived gravity vector output with low g displacement. The black line is the reference and the green line is the output.

A new test can be set up. The weighting values of the estimated gravity vector are set to 25 (they were zero before), whereas all other variables weights do not change. New outputs are represented in Figures 8.28, 8.29, 8.30, 8.31, where it is clear that the estimated gravity vector, now follows quite well its reference and remains under a smaller threshold of 0.05g. However, there is a trade-off between this achievement and the tracking of the other references. In fact, both linear accelerations and angular velocities are tracked in a worse way than the previous simulation did.

Chapter 9

Comparison with Previous Works

In previous works, [6] [14] [36], a *linear* MPC approach to the motion cueing problem is proposed. In those cases, only linear filters and linear models were used, so the overall problem remains linear and less complex than the model used in this thesis.

The aim of this chapter is to replicate the results obtained during the previous works and then compare the results with the non-linear model ones, finally find out some considerations.

9.1 Linear MPC

In previous works [6] [14] [36], it was decided to use the cascade of the platform model and vestibular system model as the final model to use in a MPC based Motion Cueing Algorithm. The inputs of the mechanical system are the longitudinal accelerations and the angular velocities applied to the platform: they represent the exogenous inputs which control the overall dynamics of the complete system. The outputs of the mechanical system (which are the inputs to the vestibular model) are the longitudinal accelerations and angular velocities the person sitting in the cockpit is subject to. The driver perceives the movements of the platform through his vestibular system, so the initial inputs are first filtered by the mechanical system, and then by the driver vestibular system. The outputs of the overall model are therefore the longitudinal accelerations and angular velocities that are perceived by the pilot. The objective of this motion cueing problem, implemented by a linear MPC, is to determine the exogenous inputs that minimize the error between the accelerations and angular velocities actually perceived and those obtained by filtering through the vestibular system the reference telemetry .

The models are expressed in a discrete-time state space representation: a discretization process is used in this implementation whereas the `ACADO toolkit` can deal directly with continuous-time models. The resulting matrices are the following:

- $A_{series} = \begin{bmatrix} A_m & 0 \\ B_{vest}C_m & A_{vest} \end{bmatrix}$
- $B_{series} = \begin{bmatrix} B_m \\ B_{vest}D_m \end{bmatrix}$
- $C_{series} = \begin{bmatrix} D_{vest}C_m & C_{vest} \end{bmatrix}$
- $D_{series} = \begin{bmatrix} D_{vest} & D_m \end{bmatrix}$

where $_m$ and $_{vest}$ indicate respectively the matrices of the mechanical and vestibular systems. These implementation yields the final model on which the MPC take place:

$$\Sigma_{series} = (A_{series}, B_{series}, C_{series}, D_{series})$$

The overall model just obtained has six degrees of freedom and is of considerable size, therefore it can be problematic to apply the MPC directly on it due to the computation time required by the predictive control. However, by carefully observing the structure of the matrices related to the mechanical and vestibular system, it is immediate to find out that the six degrees of freedom of the mechanical system are completely decoupled (the matrix A_m is diagonal), while the vestibular system can be divided into four subsystems, each one independent by the others:

- Σ_1, yaw rotations
- Σ_2, vertical accelerations
- Σ_3, the couple: longitudinal accelerations & pitch rotations
- Σ_4, the couple: lateral accelerations & roll rotations

This split is possible thanks to the linearisation introduced for the tilt coordination (same as Section 4.2) that reduce the overall system into the two independent couples: longitudinal acceleration with pitch angular position and lateral acceleration with roll angular position. In addition, it separates the dynamic of the vertical acceleration from the pitch and roll rotations.

The four subsystems here identified can be set in series to the respective mechanical subsystem and this procedure produces a total of four different systems, each one decoupled from the others. This new representation of the problem brings a lot of advantages. For example, each one of these subsystems can be implemented on a different computer thus obtaining four different controllers operating in parallel and making more simple to find the OCP solution, reducing the computation time.

9.1.1 Linear results

The test is done on the same real circuit lap, as in Section 8.3. The used data are exactly the same, so a comparison with previous simulations can be made. In particular, in Table 9.1, are reported the weighting values associated to this implementation.

Variable	Weight value $[x,y,z]$
$\vec{\hat{\omega}}$	1600, 1600, 4000
$\vec{\hat{a}}$	3000, 3000, 3000
$\vec{\theta}$	2000, 2000, 3000
$\vec{p}$	200, 200, 1000
$\vec{v}$	750, 750, 800

Variable	Weight value $[x,y,z]$
$\vec{a}$	0.0001, 0.0001, 0.0001
$\vec{\omega}$	0.00001, 0.00001, 0.1
$\vec{J}$	10, 10, 10
$\vec{\alpha}$	0.0001, 0.0001, 0.01

Table 9.1: Weighting values for the linear MPC.

where $\vec{J}$ and $\vec{\alpha}$, indicate respectively the Jerk and the angular acceleration controls.

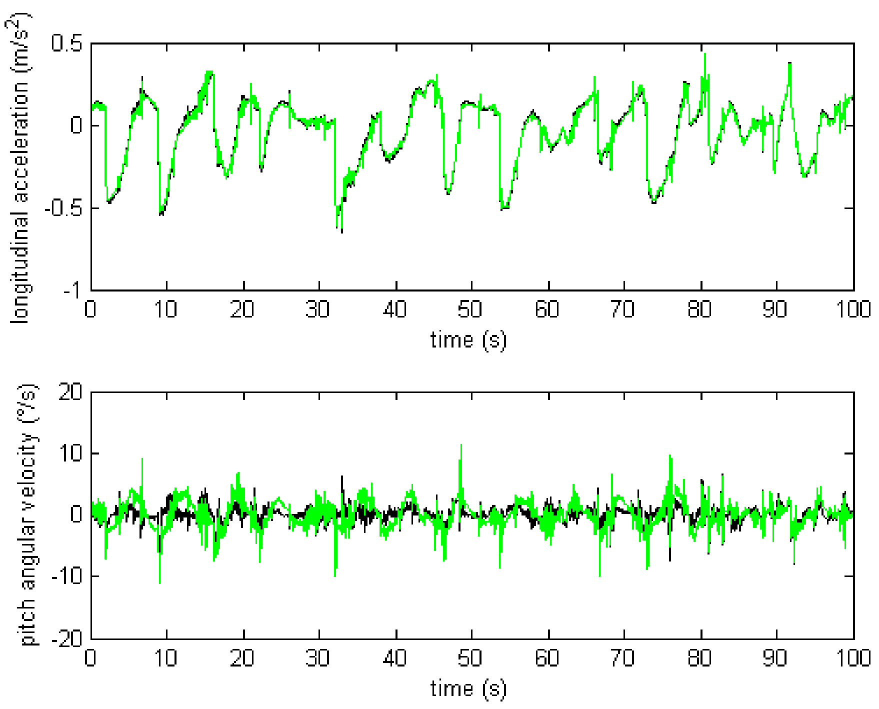


Figure 9.1: Linear MPC perceived longitudinal acceleration and pitch angular velocity outputs. The black line is the reference and the green line is the output.

In Figure 9.1, 9.2, 9.3, it is possible to see the results of this simulation. Some considerations that can be made about these results are:

- the accelerations are well tracked, even if the vertical acceleration presents some peaks that have to be avoided

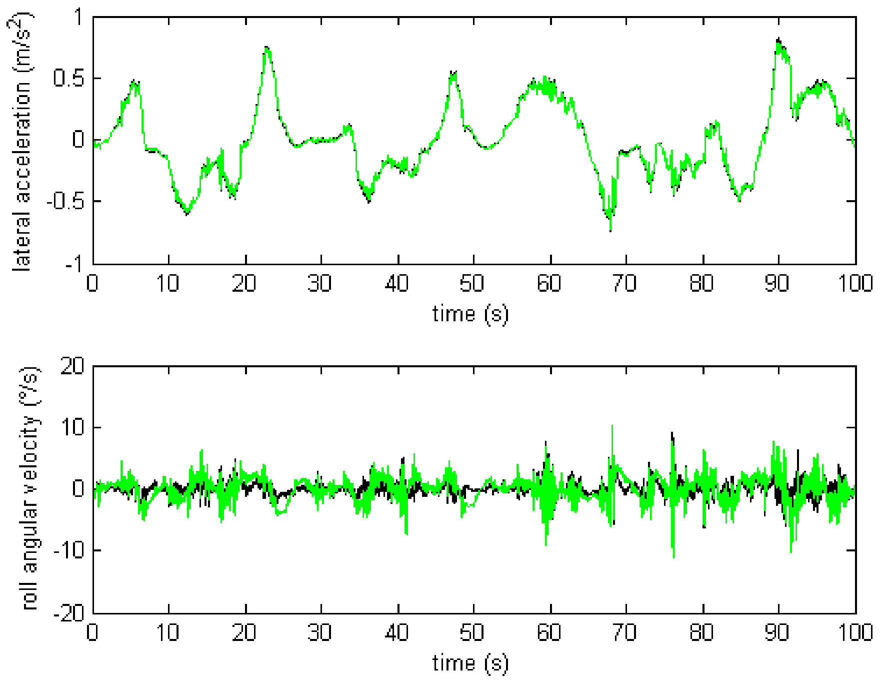


Figure 9.2: Linear MPC perceived lateral acceleration and roll angular velocity outputs. The black line is the reference and the green line is the output.

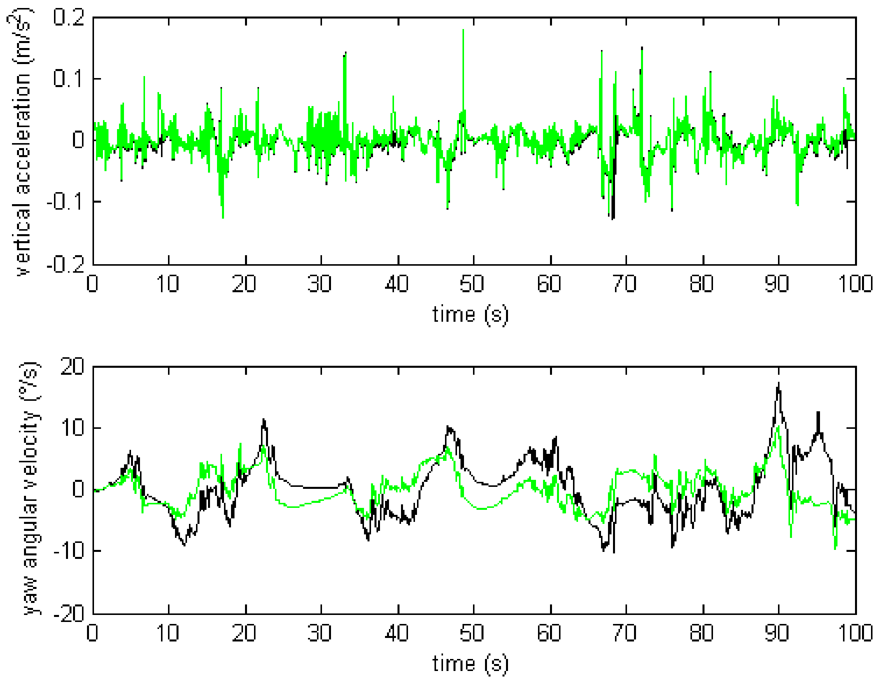


Figure 9.3: Linear MPC perceived vertical acceleration and yaw angular velocity outputs. The black line is the reference and the green line is the output.

- the angular velocities have always an acceptable magnitude with no strange behaviours. However the outputs do not track very well their respective references
- there is no information about the gravity orientation felt by the driver, so it is not possible to verify if the transmitted feelings create inadequate sensations or motion sickness.

In the end, the computation time is taken into consideration: to control and simulate a hundred seconds of data it takes about 21s. So, it is clear that the linear approach gives a big advantage on the computation cost, as was expected.

9.2 NMPC Vs linear MPC

In this section, a comparison between the MPC based on the non-linear model presented in this thesis and the linear MPC is presented. The first test is done with NMPC with fixed graphics, in this way the only difference between the two approaches is the perception model with no information given by visual cues, and then a second test with the complete non-linear model takes place. In the end some considerations are made.

First of all, it is interesting to analyse the weighting parameters of the linear MPC and to make a comparison with the respective ones of the NMPC. Weights assume subjective values, in fact, they depend upon the simulation but also upon the decisions made by the user. So, it has sense to do only some qualitative remarks looking at Tables 8.3, 8.4, 9.1: what stands out are the differences between the respective weights of the estimated linear accelerations, the estimated angular velocities and the platform tilt angles. These values are always bigger in the linear MPC than the ones applied to the NMPC, in particular, the angles by which the platform is tilted have weighting parameters bigger by almost 4 orders of size. That means the linear MPC has to pay a lot of attention about not to tilt too much the platform, whereas this problem is completely absent in the NMPC in which the non-linear model seems to have an implicit knowledge about this problem (gravity vector tilt problem).

9.2.1 References comparison

The respective references for the linear MPC and the NMPC have been generated through two totally different methods. So, it is interesting to analyse how a complex perception model, like the one used in the NMPC, could affect the generation of the references, instead of the simple filtering method used in the linear model.

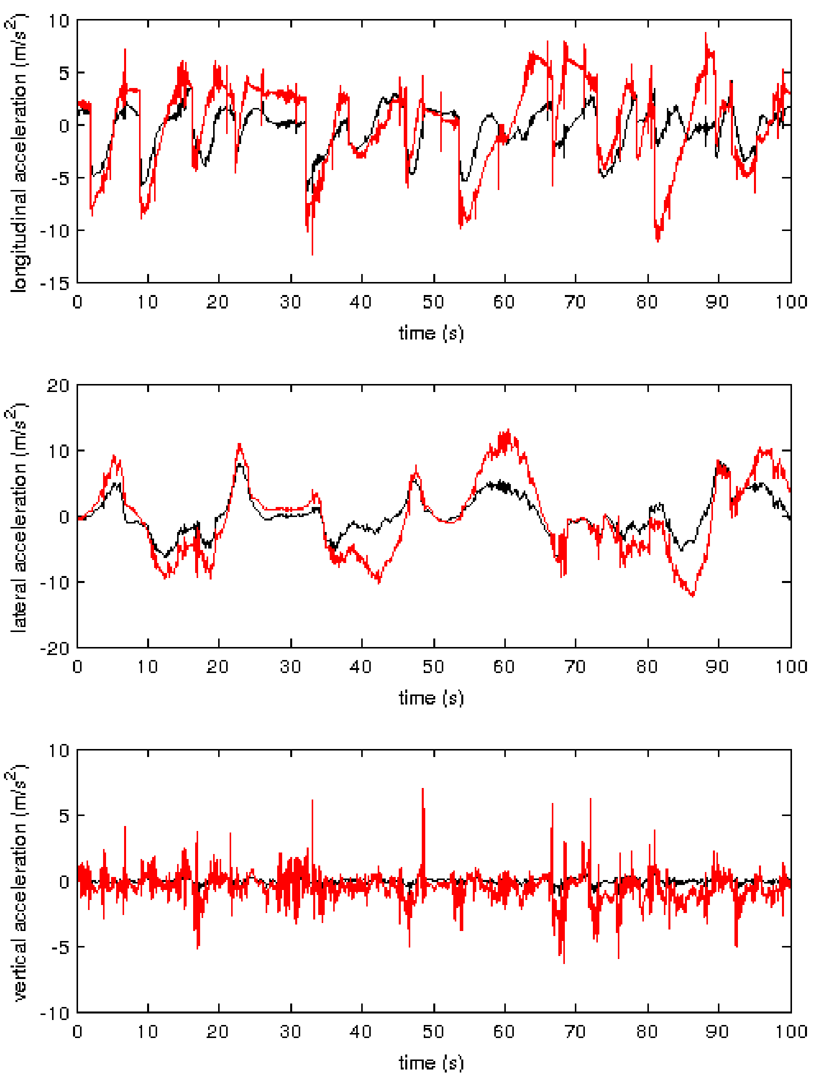


FIGURE 9.4: Comparison between the generated references for the accelerations. The black line is the reference for the linear MPC and the red line is the one for the NMPC.

Looking at Figures 9.4, 9.5, it is easy to do a fast comparison between the two different generated references and sum up some comments:

- linear accelerations are quite different, in particular, the references generated by the complex non-linear model seems to be sharper and have a greater magnitude. This behaviour can be imputed to the otolith transfer function used in this model, in fact these organs are responsible for perceived accelerations. Initially, (see Section 7.2.1), to simplify the non-linear model, the OTO transfer function is approximated to unity, while in the linear MPC is a second order rational function.

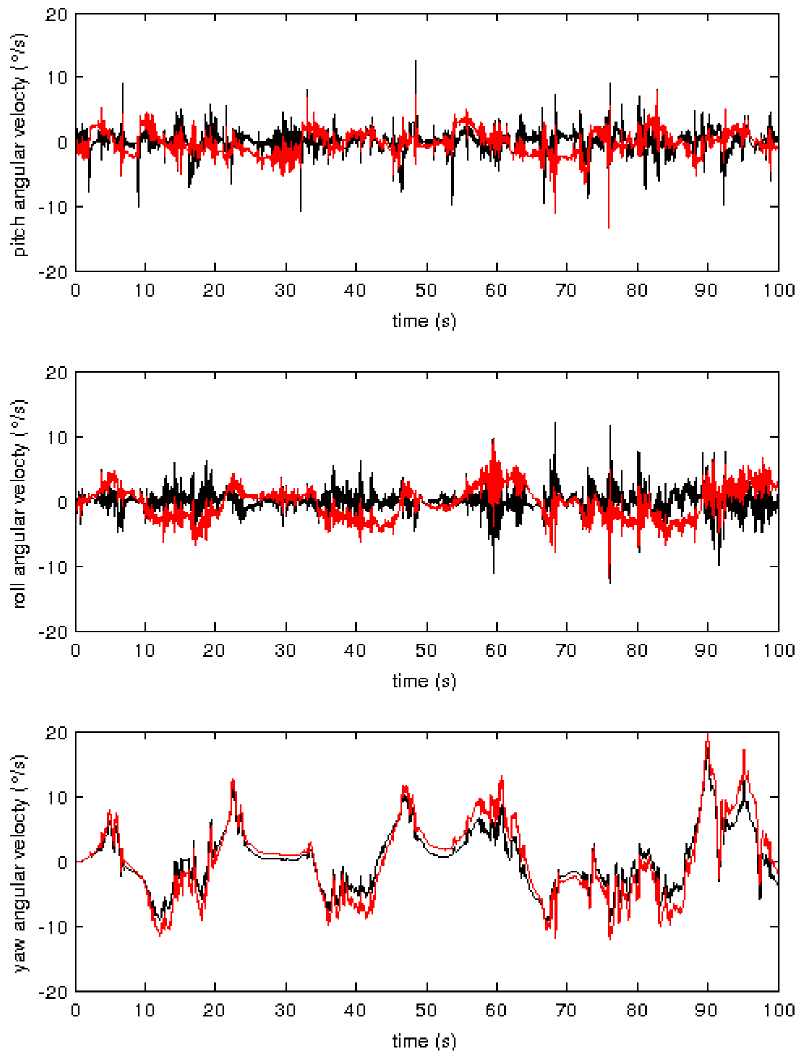


Figure 9.5: Comparison between the generated references for the angular velocities. The black line is the reference for the linear MPC and the red line is the one for the NMPC.

In this way, one reference is not filtered instead the other one is subject to an high-pass filter that takes into account only fast changes (high frequencies) of the acceleration. So the gap between the two references can be partially explained

- angular velocities are more similar, in fact the SCCs are modelled in both cases with rational transfer functions. Their dynamic is not exactly the same, in fact the linear MPC uses, also in this case, a second order rational function whereas the non-linear model implements a simpler first order transfer function. This inequality can partially explain the differences noticeable in the respective figure.

Some reasons to explain the gap between the two different generated reference has been proposed, but clearly the non-linear model has a complex dynamic and, in some cases, is hard to understand why some behaviours come out. So, the complete different approach is the best explanation for the different generated references. Obviously only one of the two can be the best, the one which gives the driver the more realistic feelings during the simulation. Then, only experimental tests on real driving simulators can give the final response.

9.2.2 Results comparison

As seen in the previous section, the generated references of the two models are not exactly the same. So, it is not consistent to do a comparison between their outputs because they will track different references. Rather, it is interesting to do a comparison of some particular behaviours, such as a strong braking or a turn, and to observe peculiar characteristics of each different approach. Only linear accelerations and angular velocities are taken into consideration because, as said before, no comparisons can be made about the gravity orientation because only the complex non-linear model gives information about that.

Fixed graphics The first comparison is done with the fixed graphics in the non-linear model, in this way the visual cues do not give any more information to the driver and so it is expected that the behaviour of the outputs could be similar to the respective ones of the linear MPC.

The first test is about how the two different approaches deal with a longitudinal braking. Clearly, the longitudinal acceleration and the pitch angular velocity are only taken into consideration. The comparison is showed in Figure 9.6 and it is possible to find out some interesting notes:

- the estimated longitudinal deceleration is almost the same during all the considered time, so the driver feelings are comparable
- the estimated pitch angular velocity, instead, has some differences. In both cases the tilt coordination is applied, in fact, the braking instant corresponds to a positive pitch angular velocity that produces a positive pitch angle. In this way, the cockpit is tilted "downward" to make the driver feel an augmented deceleration. The difference between the two implementations is a faster tilt by the NMPC that prefers to give the platform a faster acceleration trying not to move the platform too far from its neutral position.

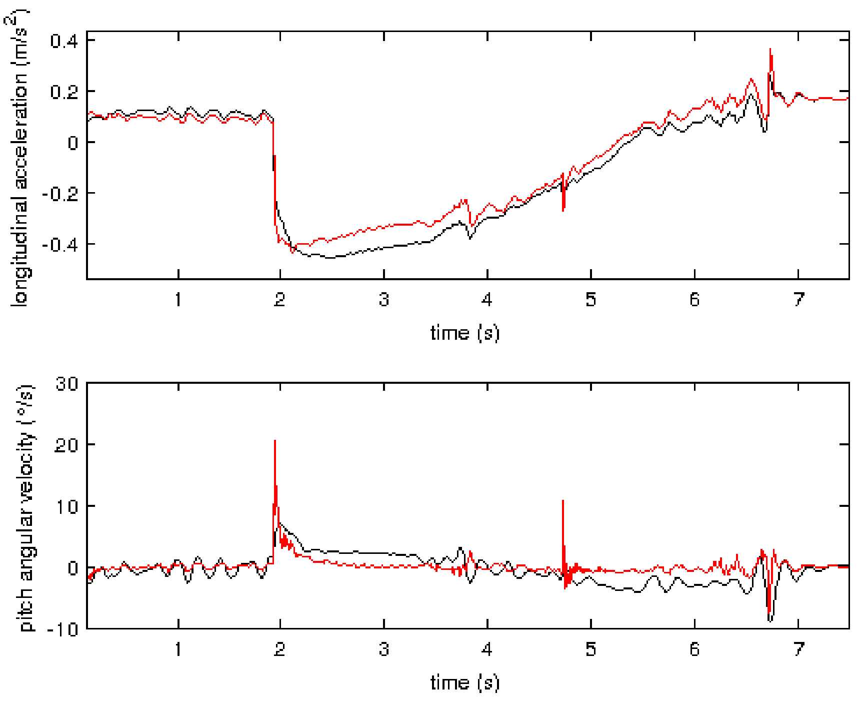


FIGURE 9.6: Different approaches to a longitudinal braking. The black line is the the linear MPC output and the red line is the one of the NMPC with fixed graphics.

This behaviour has pros & cons, in fact, as said, the platform has to do a smaller displacement that goes in advantage to future movements, whereas a higher and sharper angular velocity can cause problems like motion sickness.

The second test deals with the the first right turn of the lap that happens just after the deceleration of the previous test. Here the estimated yaw angular velocity and the platform yaw angle are taken into consideration. Figure 9.7 shows this test comparison.

- the estimated angular velocities are very similar in the first part, when the car is going almost straight ahead, but when it comes the right turn (4-7s) the NMPC estimated yaw angular velocity maintains a smoother trend
- the platform yaw angles, instead, are quite different. The NMPC commands the platform to perform a bigger positive yaw angle that is not immediately recovered by rotating back the platform to its neutral position, thing that the linear MPC clearly does in advance.

Enabled graphics In this paragraph, all the tests done with the NMPC with fixed graphics are repeated, but this time with the graphics control enabled. So it is possible

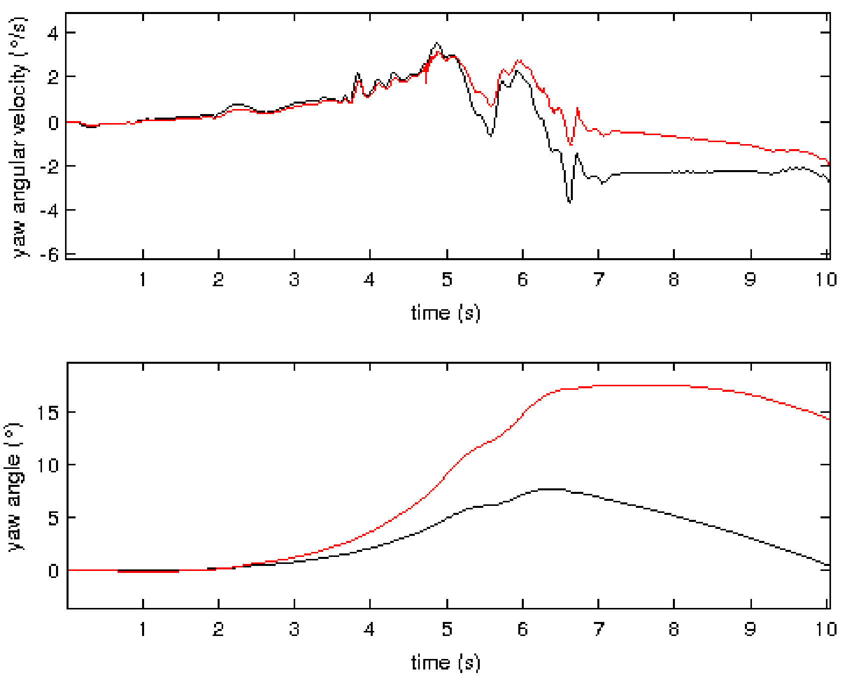


FIGURE 9.7: Different approaches to a right turn. The black line is the the linear MPC output and the red line is the one of the NMPC with fixed graphics.

to find out how these 3 more DOFs influence the feelings of the driver and the platform movements.

The braking test comparison is showed in Figure 9.8 and it is possible to make some comments:

- as in the previous tests, the estimated longitudinal accelerations are very similar, the feelings of the driver are probably comparable
- looking at the angular velocities picture in Figure 9.8, a strange behaviour is clearly reproduced. But it can be easily explained by the visual angular velocity control (see Figure 9.9). In fact, the perception of a negative pitch angular velocity is achieved by the false visual cues whereas the platform is performing a positive pitch rotation to obtain the right tilt coordination to reproduce the deceleration. Figure 9.9 clearly shows how the visual pitch angular velocity follows the platform pitch rotation and even reaches an higher value, so the driver feels an opposite angular velocity than what is really taking place.

The right turn test comparison is showed in Figure 9.10.

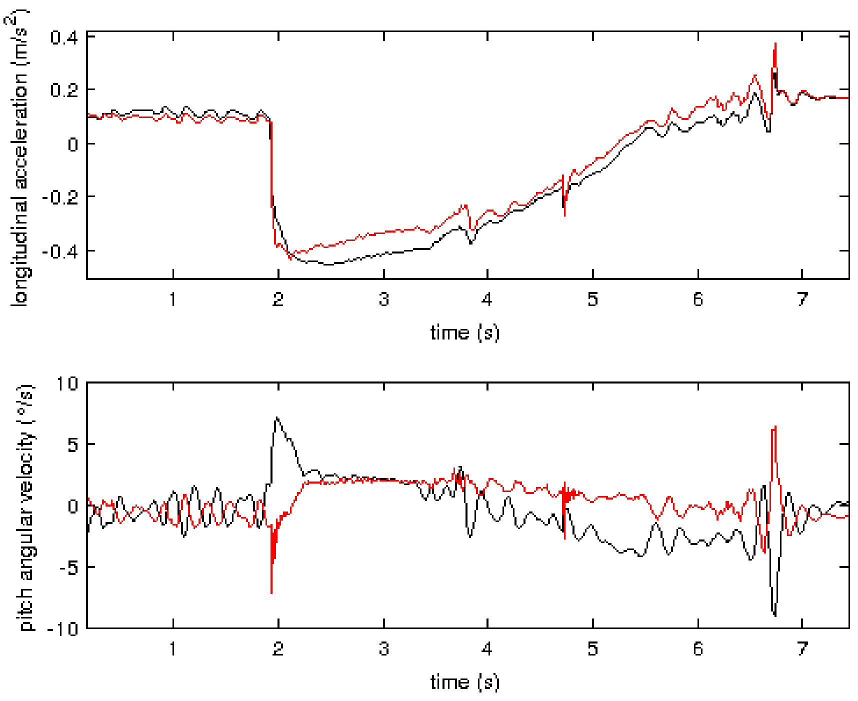


FIGURE 9.8: Different approaches to a longitudinal braking. The black line is the the linear MPC output and the red line is the one of the NMPC with enabled graphics.

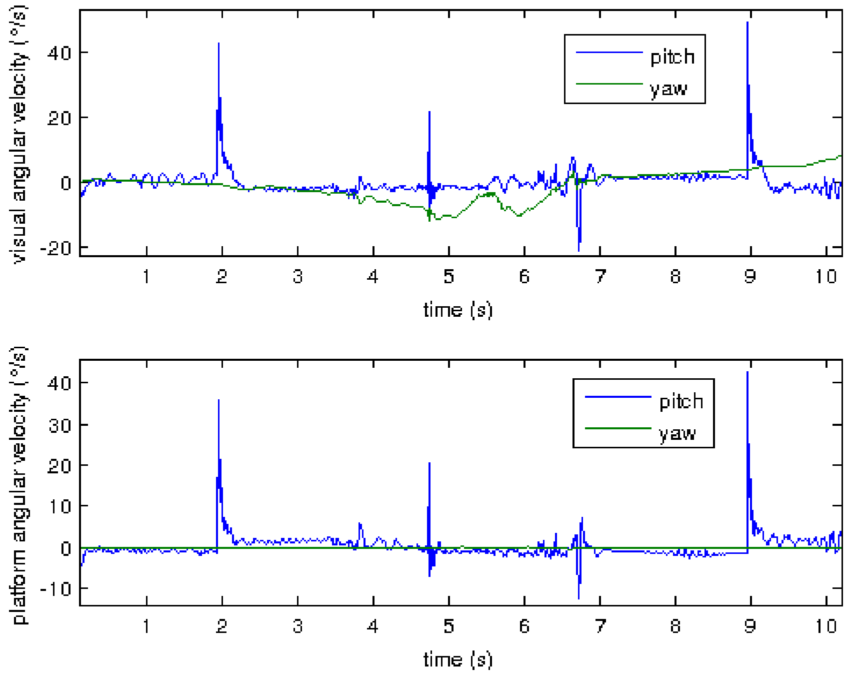


FIGURE 9.9: Visual and platform pitch and yaw angular velocities during braking and right turn.

- yaw angular velocities have a similar trend but the NMPC output is always a bit larger than the one produced by the linear MPC; it is not much relevant
- the computed platform yaw angles, instead, are very different. During the right turn, the platform controlled by the NMPC even does not rotate along the z axis, while, as before, the linear MPC imposes a significant yaw angle. Clearly the feelings of the driver are very similar and this demonstrates that, in this case, the non-linear controller has preferred to achieve the feeling of rotation using, again, the visual control (see Figure 9.9). In fact, from the fourth second of simulation, a negative yaw visual angular velocity is performed, as a consequence the driver feels a positive yaw rotation even if the platform is not rotating. Then the visual control becomes positive to recover the neutral position, that corresponds exactly to the behaviour of the linear MPC.

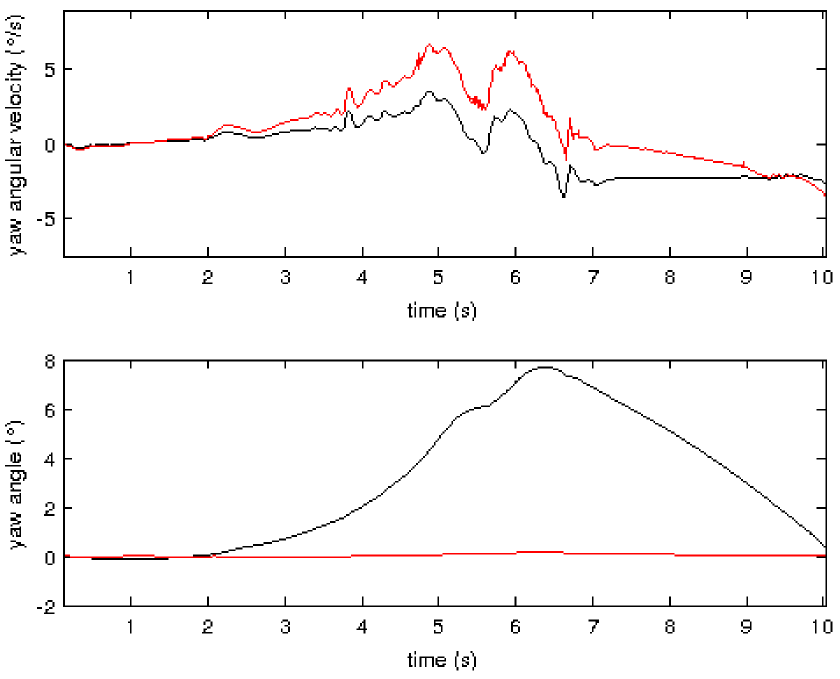


FIGURE 9.10: Different approaches to a right turn. The black line is the the linear MPC output and the red line is the one of the NMPC with enabled graphics.

So, it is clear that the visual control has a heavy impact in the overall behaviour of the NMPC. However, the magnitude of the visual angular velocity was not limited during these tests and it is not known if the values it reaches are too high and could cause some problems like motion sickness. Anyway, it is always possible to set a bigger weight or a bound on this control during the set-up of the OCP.

Chapter 10

Conclusions and future works

In this thesis was developed a multi-sensory observer model for human spatial orientation perception. The model was successfully implemented and used to predict the responses to a number of visual and visual-vestibular motion paradigms and, moreover, it was used to simulate a driving session to test an actual application. The vestibular aspects of the model derive from Merfeld's 1993 multidimensional sensory conflict model [21] and subsequent structural and parameter value refinements by Merfeld and Zupan [26]. Further extensions were required to obtain the necessary state estimates for visual sensory interaction and to capture the qualitative characteristics of large physical scale integrated self motion.

Visual information is derived from Newman's multi-sensory observer model [28]. It was added to the extended vestibular model consistently with the Merfeld (1993) topology. Two visual sensors were hypothesized to process both static (gravity/"down") and dynamic (angular velocity) visual input. The visual gravity was supposed to depend on the visual angular velocity, namely the angular velocity was integrated to obtain the rotation angles of the graphics and these values were used to compute the visual gravity input using spherical coordinates. In the complete model, every sensory output was compared with the respective expected value from an internal CNS model and the resultant sensory conflict signal was weighted and added to the rate of change of the respective state.

As expected, the model predicted accurate perceptual responses for the experimental simulations considered and validated by Merfeld and showed to have an implicit knowledge about the tilt coordination, in fact it used this mechanism in a totally automatic way. This quality is clearly shown in the real circuit lap tests, during which very good results were presented.

Although the model was successful in meeting the prefixed goals, several limitations exist and require additional attention and refinement. For example, sensory noise could be added and used to model perceptual thresholds of the semicircular canals and otolith organs. These thresholds are very important because they establish the border between what the driver does perceive or does not through his vestibular system. Thresholds about the gravity perception must also be investigated and can deeply influence the set up of the model. Additional visual system non-linearities and dynamics could be incorporated to account for saturation limits, vection delays and other characteristics of focal and periphery vision.

Then it comes the real-time implementation problem. In fact, the final purpose of these tests is to set up the NMPC on a real driving simulator. Clearly this application requires a real-time implementation, that has to work at 100Hz without problems. The presented results did not respect this condition even if they gave interesting hints for the future works. An idea, that will surely improve the computation time of the algorithm, is to implement the code directly using the `ACADO toolkit` (`C/C++` code) without taking advantage of its Matlab interface.

Finally, several pertinent experiments are required to further validate the model structure and implementation, especially some driving simulator sessions would be very useful to sum up the driver's feelings and compare them with the different perceived sensations that can be achieved with other available approaches to the motion cueing problem.

Appendix A

Coordinate systems and spatial rotations

A.1 Coordinate systems

It is defined a right handed coordinate system relative to the world (X_w, Y_w, Z_w) and the head (X, Y, Z), Figure A.1. It is assumed that the semicircular canals and otolith organs are situated at the center of the head and aligned with the naso-occiptal X, interaural Y, and dorsoventral Z, axes. For computational simplicity angular velocity and linear acceleration inputs to the vestibular model are processed in the egocentric head fixed frame. It is often necessary to transform quantities between reference frames. Gravity, for instance, is inherently defined in world coordinates yet needed for the GIF ("gravito-inertial force") calculation performed in the head axes. As we rotate and translate about in space a novel description of the relationship between these coordinate frames is therefore required.

A.2 Quaternion representation

The quaternion provides a useful notation for representing spatial rotations. Quaternions eliminate gimbal lock, reduce numerical storage from 9 to 4 digits (9 being the typical representation of a rotation matrix), and increase computational stability. A quaternion representation for the coordinate frames and vectorial rotations is therefore preferred to rotation matrices.It is possible to define a unit quaternion in the following form:

$$\overrightarrow{q} = q_0 + q_1 i + q_2 j + q_3 k$$

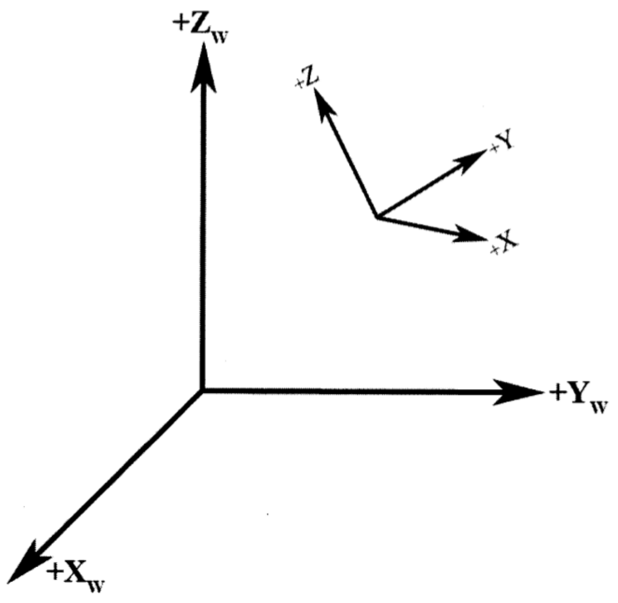


FIGURE A.1: Head and World coordinate frame.

$$\text{with} \quad i^2 = j^2 = k^2 = -1$$

In order to update the quaternion vector as a rotation in the inertial space take place, the initial quaternion $\overrightarrow{q}_0$ must be integrated with respect to the angular velocity input, $\overrightarrow{\omega}(t) = [\omega_x(t),\ \omega_y(t),\ \omega_z(t)]^T$. A stable algorithm to perform this integration was developed by Fang & Zimmerman (1969).

$$\dot{q}_0 = -\frac{1}{2}(q_2\omega_x + q_1\omega_y + q_3\omega_z) + k\lambda q_0$$

$$\dot{q}_1 = -\frac{1}{2}(q_1\omega_x - q_0\omega_y - q_2\omega_z) + k\lambda q_1$$

$$\dot{q}_2 = \frac{1}{2}(q_0\omega_x + q_3\omega_y - q_1\omega_z) + k\lambda q_2$$

$$\dot{q}_3 = \frac{1}{2}(q_1\omega_x - q_2\omega_y + q_0\omega_z) + k\lambda q_3$$

$$\lambda = 1 - (q_0^2 + q_1^2 + q_2^2 + q_3^2)$$

Integrating these equations yields a complete time history for the quaternion vector $\overrightarrow{q}$. This particular formulation uses an algebraic constraint to minimize the constraint error. For alternate integration schemes using normalization or derivative constraints one is referred to Fang & Zimmerman [37]. Constraint errors represent a non-orthonormality in the transformation matrix and are thus extremely problematic for the decomposition of vectors. The proportionality constant "k" ensures stability such that $k > 0$ and the product $hk < 1$, where h is defined as the integration time step.

The integrated quaternion now provides all the necessary information to transform vectors between the head and world coordinate frames. At each time step a rotation of the gravity vector $\overrightarrow{g}_w = [g_{xw},\ g_{yw},\ g_{zw}]^T$ is accomplished with the transformation matrix T;

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_0q_1 + q_2q_3) \\ 2(q_0q_2 + q_1q_3) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

such that the current direction of gravity in head coordinates can be expressed as the premultiplication of $\overrightarrow{g}_w$ with T.

$$\overrightarrow{g}(t) = R(\overrightarrow{q})\overrightarrow{g}_w$$

Using this relationship and its inverse, it is possible to simply obtain the linear acceleration $\overrightarrow{a}_w$ and angular velocity $\overrightarrow{\omega}_w$ controls of the platform from the head-fixed driver frame as:

$$\overrightarrow{a}_w = R^{-1}\overrightarrow{a} = R^T\overrightarrow{a}$$

$$\overrightarrow{\omega}_w = R^{-1}\overrightarrow{\omega} = R^T\overrightarrow{\omega}$$

Appendix B

ACADO Toolkit

ACADO Toolkit is a software environment and algorithm collection for automatic control and dynamic optimization. It provides a general framework for using a great variety of algorithms for direct optimal control, including model predictive control, state and parameter estimation and robust optimization. ACADO Toolkit is implemented as self-contained C++ code and comes along with user-friendly MATLAB interface. The object-oriented design allows for convenient coupling of existing optimization packages and for extending it with user-written optimization routines.

Every useful information can be found at [38].

B.1 Features

ACADO Toolkit is designed to meet some key properties that are useful in this thesis work.

Non-linear optimal control One of the basic problem classes which can be solved with ACADO toolkit are standard optimal control problems [1]. These problems typically consist of a dynamic system with differential states and possibly also algebraic states, the objective can usually be written as a sum of a Lagrange and a Mayer term. Moreover, ACADO toolkit tackles several types of constraints, such as control and state bounds, terminal constraints, general non-linear path constraints, periodic boundary conditions, etc.

Model predictive control Another highlight of ACADO Toolkit are its model based feedback control algorithms. The corresponding problems can be divided into two kinds of online dynamic optimization problems: the Model Predictive Control (MPC) problem of finding (approximately) optimal control actions to be fed back to the controlled process, and the Moving Horizon Estimation (MHE) problem of estimating the current process states using measurements of its outputs.

Code Generation for Fast NMPC and MHE The ACADO Code Generation tool can automatically generate Gauss-Newton real-time iteration algorithms for fast non-linear MPC and MHE applications. Based on the symbolic syntax of the ACADO Toolkit, it allows the user to export highly efficient and self-contained C code that is tailored to each respective MPC or MHE problem formulations. Computational speed is increased by hard-coding all problem dimensions, avoiding dynamic memory allocations, loop unrolling, symbolic simplifications and the use of a fixed-step integrator. This can lead to significant speed-ups compared to generic implementations.

B.2 QP Problem

A linearly constrained optimization problem with a quadratic objective function is called a quadratic program (QP). Because of its many applications, quadratic programming is often viewed as a discipline in and of itself. More importantly, though, it forms the basis of several general non-linear programming algorithms. We begin this section by examining the Karush-Kuhn-Tucker conditions for the QP and see that they turn out to be a set of linear equalities and complementarity constraints. Much like in separable programming, a modified version of the simplex algorithm can be used to find solutions.

Problem statement The general quadratic program can be written as

$$\text{Minimize} \;\; f(x) = cx + \frac{1}{2}x^T Qx$$

$$\text{subject to} \;\; Ax \leq b \; and \; x \geq 0$$

where c is an n-dimensional row vector describing the coefficients of the linear terms in the objective function, and Q is an $n \times n$ symmetric matrix describing the coefficients of the quadratic terms. If a constant term exists it is dropped from the model. As in linear programming, the decision variables are denoted by the n-dimensional column vector x, and the constraints are defined by an $m \times n$ A matrix and an m-dimensional column vector b of right-hand-side coefficients. We assume that a feasible solution exists and that the constraint region is bounded.

When the objective function $f(x)$ is strictly convex for all feasible points the problem has a unique local minimum which is also the global minimum. A sufficient condition to guarantee strictly convexity is for Q to be positive definite

Karush-Khun-Tucker conditions We now specialize the general first-order necessary conditions to the quadratic program. These conditions are sufficient for a global minimum when Q is positive definite; otherwise, the most we can say is that they are necessary.

Excluding the nonnegativity conditions, the Lagrangian function for the quadratic program is

$$L(x,\mu) = cx + \frac{1}{2}x^TQx + \mu(Ax-b)$$

where μ is an m-dimensional row vector. The Karush-Kuhn-Tucker conditions for a local minimum are given as follows.

$$\begin{array}{lcl}
\dfrac{\partial L}{\partial x_j} \geq 0,\ j = 1.\dots,n & \Longrightarrow & c + x^TQ + \mu A \geq 0 \\
\dfrac{\partial L}{\partial \mu_i} \leq 0,\ i = 1.\dots,m & \Longrightarrow & Ax - b \leq 0 \\
x_j\dfrac{\partial L}{\partial x_j} = 0,\ j = 1.\dots,n & \Longrightarrow & x^T(c^T + Qx + A^T\mu) = 0 \\
\mu_i g_i(x) = 0,\ i = 1.\dots,m & \Longrightarrow & \mu(Ax-b) = 0 \\
x_j \geq 0,\ j = 1.\dots,n & \Longrightarrow & x \geq 0 \\
\mu_i \geq 0,\ i = 1.\dots,n & \Longrightarrow & \mu \geq 0
\end{array}$$

Appendix C

Graphs

Here are shown some more graphs about all the tests done in this thesis and that were not reported directly in their respective section due to space reasons.

NMPC with fixed graphic Some graphs about the platform linear position and angular position during the NMPC test with fixed graphic.

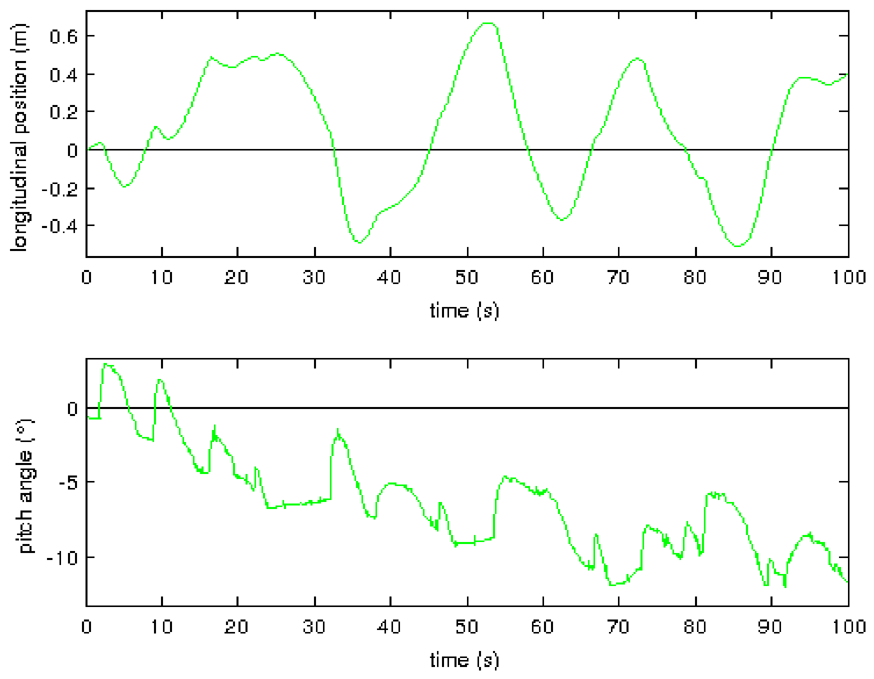


FIGURE C.1: NMPC longitudinal position and pitch angular position output with fixed graphic. The black line is the reference and the green line is the output.

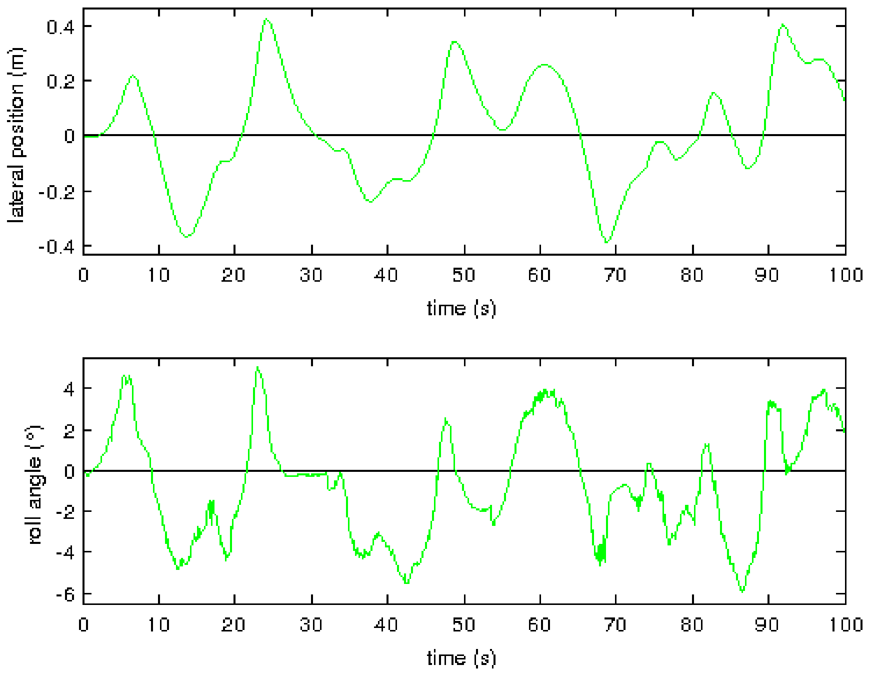


FIGURE C.2: NMPC lateral position and roll angular position output with fixed graphic. The black line is the reference and the green line is the output.

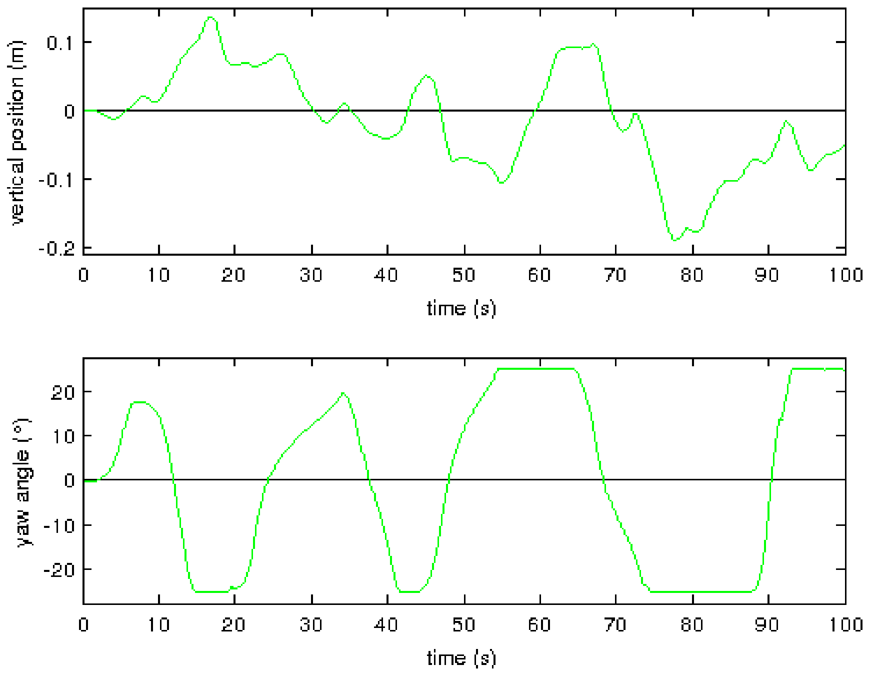


FIGURE C.3: NMPC vertical position and yaw angular position output with fixed graphic. The black line is the reference and the green line is the output.

NMPC with enabled graphic Some graphs about the platform linear position and angular position during the NMPC test with enabled graphic.

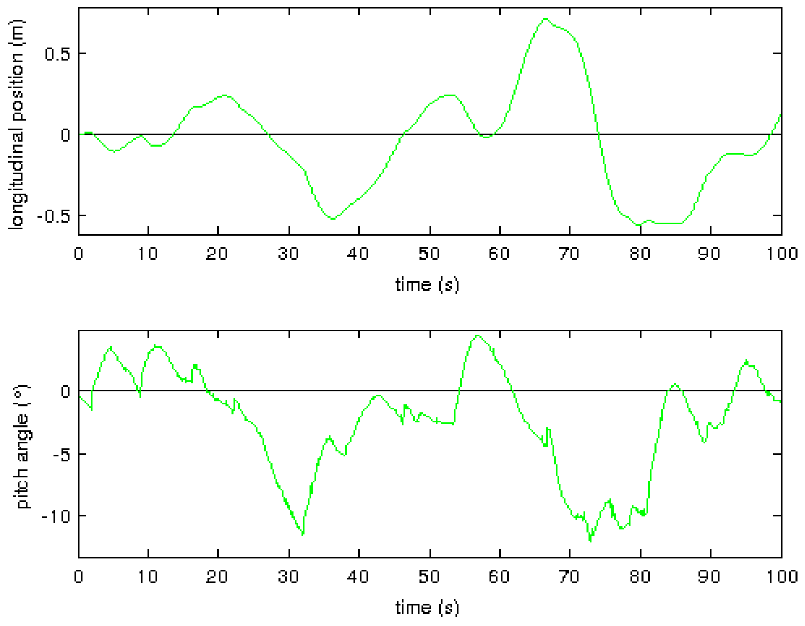


FIGURE C.4: NMPC longitudinal position and pitch angular position output with enabled graphic. The black line is the reference and the green line is the output.

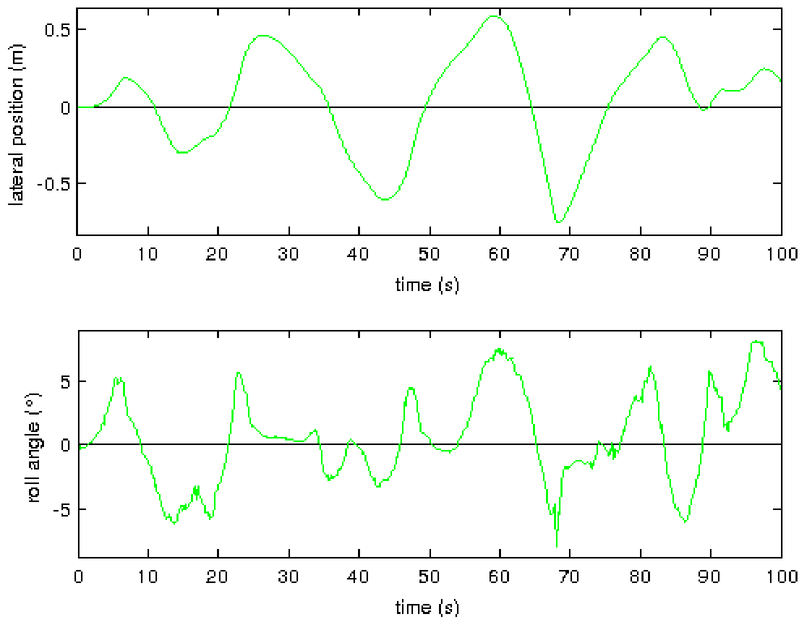


FIGURE C.5: NMPC lateral position and roll angular position output with enabled graphic. The black line is the reference and the green line is the output.

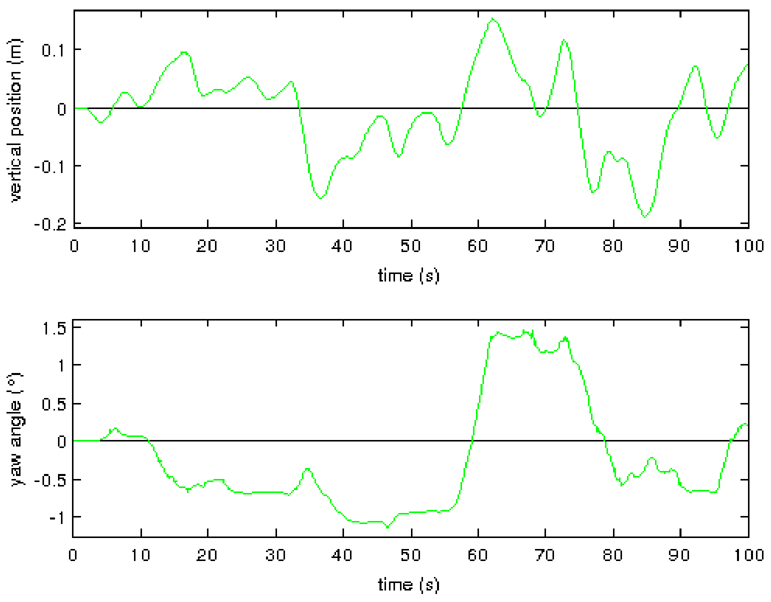


FIGURE C.6: NMPC vertical position and yaw angular position output with enabled graphic. The black line is the reference and the green line is the output.

NMPC with enabled graphic and low g displacement Some graphs about the platform linear position and angular position during the NMPC test with enabled graphic and low g displacement.

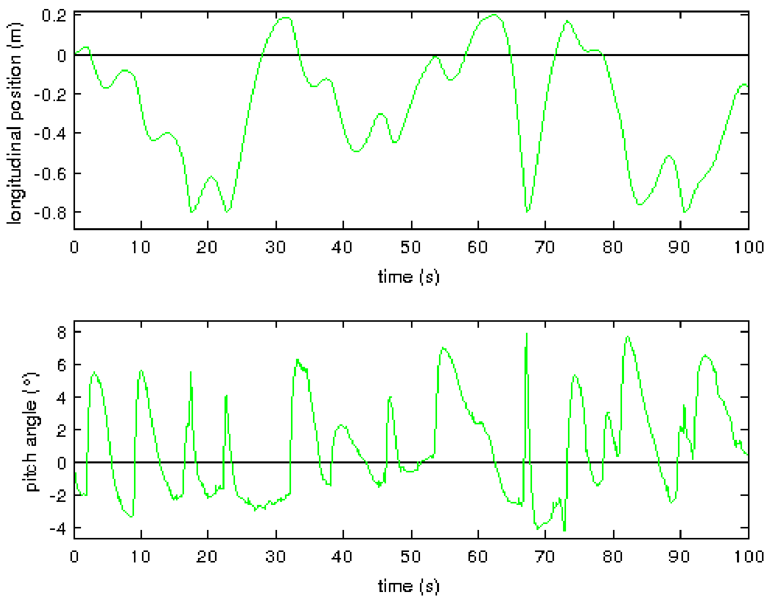


FIGURE C.7: NMPC longitudinal position and pitch angular position output with low g displacement. The black line is the reference and the green line is the output.

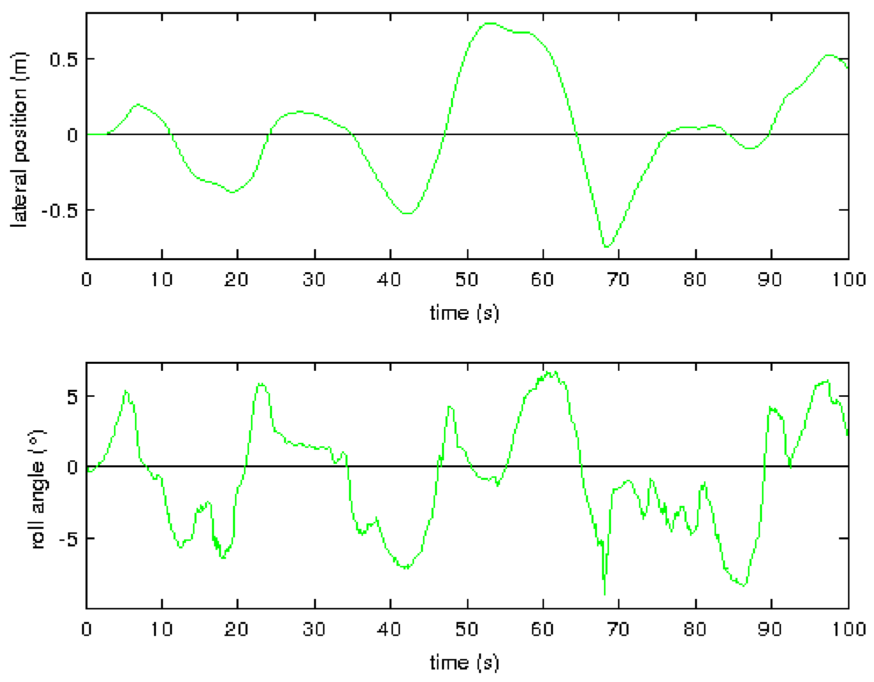


FIGURE C.8: NMPC lateral position and roll angular position output with low g displacement. The black line is the reference and the green line is the output.

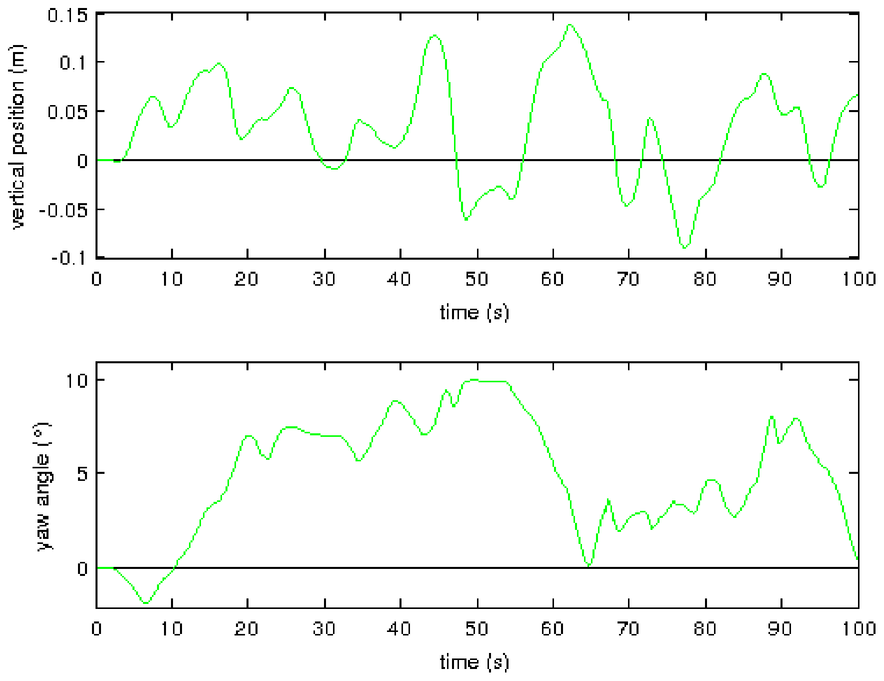


FIGURE C.9: NMPC vertical position and yaw angular position output with low g displacement. The black line is the reference and the green line is the output.

Linear MPC Some graphs about the platform linear position and angular position during the linear MPC test.

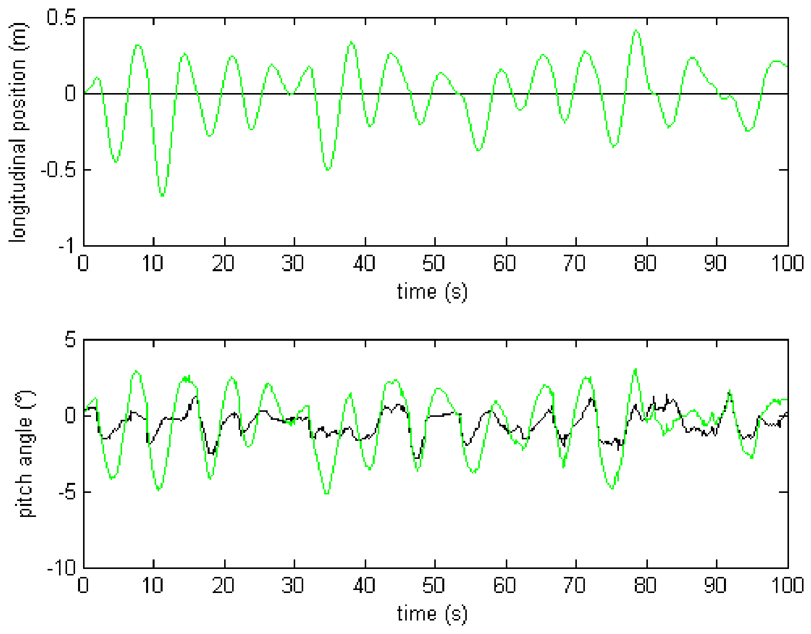


FIGURE C.10: Linear MPC longitudinal position and pitch angular position output. The black line is the reference and the green line is the output.

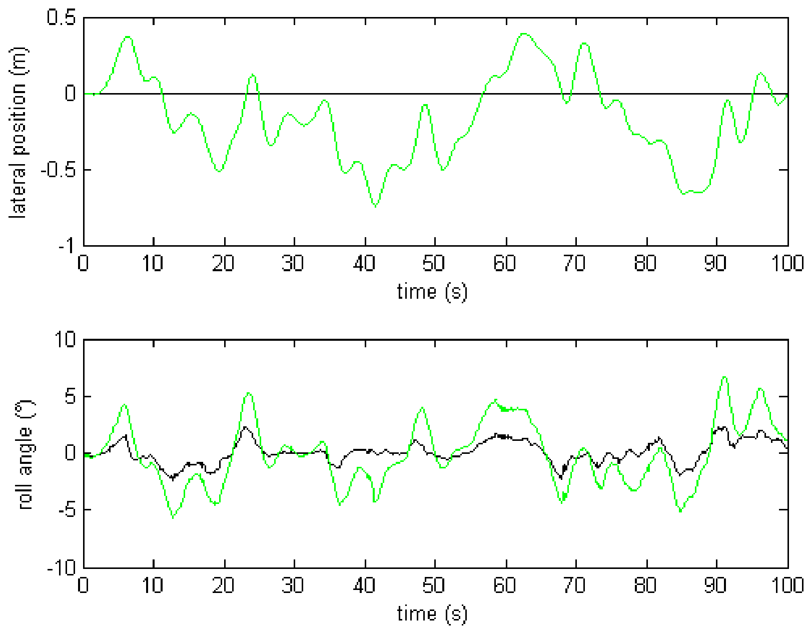


FIGURE C.11: Linear MPC lateral position and roll angular position output. The black line is the reference and the green line is the output.

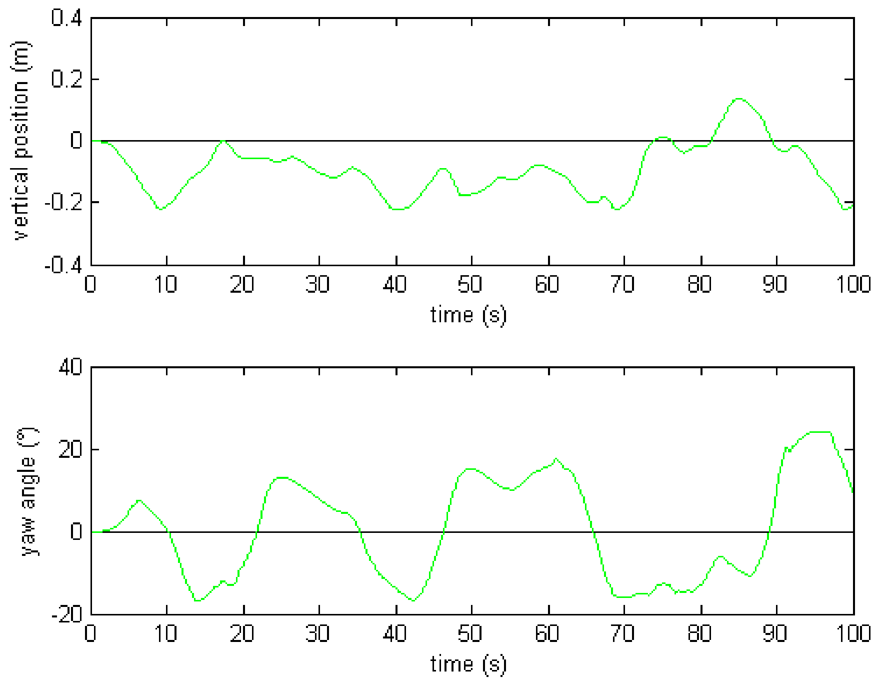


FIGURE C.12: Linear MPC vertical position and yaw angular position output. The black line is the reference and the green line is the output.

Bibliography

[1] P. Selva. Modeling of the vestibular system and nonlinear models for human spatial orientation perception. *Ph.D Thesis, Institut Supérieur de l'Aéronautique et de l'Espace, Universitè de Toulouse*, 2009.

[2] M.G. Braithwaite, S.J. Durnford, S.L. Groh, H.D. Jones, A.A Higdon, A. Estrada, and E.A. Alvarez. *Flight simulator evaluation of a novel flight instrument display to minimize the risks of spatial disorientation.* 1998.

[3] P.R. MacNeilage, N. Ganesan, and D.E. Angelaki. Computational approaches tp spatial orientation: from transfer functions to dynamic bayesian inference. *Journal of Neurophysiology*, 2008.

[4] URL http://www.amm.mw.tum.de/.

[5] D. D'Ambrosio. Algoritmi di motion cueing per simulatori di veicolo. *Tesi di Laurea Magistrale in Ingegneria dell'Automazione, Università di Padova*, 2010.

[6] M. Baseggio. Generazione dei riferimenti per algoritmi di motion cueing basati su mpc. *Tesi di Laurea Magistrale in Ingegneria dell'Automazione, Università di Padova*, 2011.

[7] L.D. Reid and M.A. Nahon. Flight simulation motion-based drive algorithms: part 1 - developing and testing the equations. 1985.

[8] G. Reymond and A. Kemeny. Motion cueing in the renault driving simulator. 2000.

[9] R.V. Parrish and J.E. Dieudonne. Coordinated adaptive washout for motion simulators. *Journal of Aircraft*, 1975.

[10] R. Sivan and J. Ish Shalom. An optimal control approach to the design of moving flight simulators. *IEEE International Conference on Systems, Man and Cybernetics*, 1982.

[11] L. Wang. *Model predictive control system design and implementation using MATLAB.* 2009.

[12] J.M. Maciejowski. *Predictive control with constraints.* 2002.

[13] B.D.C. Augusto and R.J.L. Loureiro. Motion cueing in the chalmers driving simulator: A model predictive control approach. *Master of Science Thesis, Chalmers University of Technology*, 2009.

[14] M. Pozzi. Motion cueing per simulatori dinamici di veicolo: algoritmi mpc basati su modelli percettivi. *Tesi di Laurea Magistrale in Ingegneria dell'Automazione, Università di Padova*, 2011.

[15] J.P. Sauvage. *Anatomie de l'oreille interne.* 1999.

[16] R.D. Rabbitt, E.R. Damiano, and J.W. Grant. *Biomechanics of the semicircular canals and otolith organs.* 2004.

[17] R.D. Rabbit. Directional coding of three-dimensional movements by the vestibular semicircular canals. *Biol. Cybern.*, 1999.

[18] I.S. Curthoys and C.M. Oman. Dimensions of the horizontal semicircular duct, ampulla, and utricle in the human. *Acta Otolaryngol.*, 1987.

[19] URL http://en.wikipedia.org/wiki/Nystagmus.

[20] D.M. Merfeld. Spatial orientation in the squirrel monkey: an experimental and theoretical investigation. *Doctor of Philosophy Thesis, Massachusetts Institute of Technology*, 1990.

[21] D.M. Merfeld, L.R. Young, C.M. Oman, and M.J. Shelhamer. A multidimensional model of the effects of gravity on the spatial orientation of the monkey. *Journal of Neurophysiology*, 1993.

[22] L R. Young. On visual-vestibular interaction. *Proc. 5th Symposium of the Role of the Vestibular Organs in Space Exploration. NASA*, 1970.

[23] C M. Oman. A heuristic mathematical model for the dynamics of sensory conflict and motion sickness. *Acta Oto. Suppl.*, 1999.

[24] J.A. Goldberg and C. Fernandez. Physiology of peripheral neuronons innervating semicircular canals of the squirrel monkey. *J. of Neurophysiology*, 1971.

[25] C Fernandez and J. Goldberg. Physiology of peripheral neurons innervating the otolith organs of the squirrel monkey. *J. of Neurophysiology*, 1976.

[26] D.M. Merfeld and L.H. Zupan. Neural processing of gravitoinertial cues in humans. iii modeling tilt and translation responses. *Journal of Neurophysiology*, 2002.

[27] M.C. Newman. A multisesnory observer model for human spatial orientation perception. *Master of Science in Aeronautics and Astronautics, Massachusetts Institute of Technology*, 2009.

[28] M.C. Newman, B.D. Lawson, A.H. Rupert, and B.J. McGrath. The role of perceptual modeling in the understanding of spatial disorientation during flight and ground-based simulator training. *AIAA Modeling and Simulation Technologies Conference*, 2012.

[29] J. Borah, L.R. Young, and R.E. Curry. *Optimal Estimator Model for Human Spatial Orientation. Representation of Three-Dimensional Space in the Vestibular, Oculomotor, and Visual System.* 1988.

[30] T. Haslwanter, R. Jaeger, S. Mayr, and M. Fetter. *Three-dimensional Eye-movement Responses to Off-vertical Axis Rotations in Humans.* 2000.

[31] S.J. Wood, M.F. Reschke, L.A. Sarmiento, and G. Clement. Tilt and translation motion perception during off-vertical axis rotation. *Experimental Brain Research*, 2007.

[32] G.M. Jones and L.R. Young. Subjective detection of vertical acceleration: A velocity dependent response? *Acta Otolaryngologica*, 1978.

[33] URL `http://www.real-world-physics-problems.com/`.

[34] B. Cohen, V. Matsuo, and T. Raphan. Quantitative analysis of the velocity characteristics of optokinetic nystagmus and optokinetic agter-nystagmus. *Journal of Physiology*, 1977.

[35] C. Darlot, P. Denise, J. Droulez, B. Cohen, and A. Berthoz. Eye movements induced by off-vertical axis rotation (ovar) at small angles of tilt. *Experimental Brain Research*, 1988.

[36] A. Beghi, M. Bruschetta, and F. Maran. A real time implementation of mpc based motion cueing strategy for driving simulators. *Conference on Decision and Control*, 2012.

[37] A.C. Fang and B.G. Zimmerman. Digital simulation of rotational kinematics. *Goddard Space Flight Center; NASA TN D-5302*, 1969.

[38] URL `http://acado.github.io/`.