Universidad
Carlos III de Madrid

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

TESIS DE MÁSTER

EQUILIBRIUM CONTROL FOR HUMANOID ROBOT TEO BY INERTIAL PERCEPTION

Autor: Juan Lorente Monzó
Tutor: Juan Miguel García Haro
Director: Santiago Martínez de la Casa Díaz

MÁSTER OFICIAL EN
ROBÓTICA Y AUTOMATIZACIÓN

LEGANÉS, MADRID
JUNIO 2016

UNIVERSIDAD CARLOS III DE MADRID
MÁSTER OFICIAL EN ROBÓTICA Y AUTOMATIZACIÓN

El tribunal aprueba la Tesis de Máster titulada "**Equilibrium Control for Humanoid Robot TEO by Inertial Perception**" realizada por **Juan Lorente Monzó**.

Fecha: Junio 2016

Tribunal:

_____
Dr./Dra.

_____
Dr./Dra.

_____
Dr./Dra.

"A guy can change anything: his face, his home,
his family, his girlfriend, his religion, his God...
But there's one thing he can't change.
He can't change his PASSION."
The Secret In Their Eyes

Contents

Index of Tables

Index of Figures

Acknowledgments

Firstly, thanks a lot to Santiago Martínez, who made possible for me to engage this interesting project and who advised me through all the way. I am also very grateful to Juan G. Victores, Juan Miguel García, Jorge Muñoz and Santiago Morante because they didn't hesitate to help me whenever I needed. And of course thanks to all my laboratory collegues. They made my time there much more fun and enjoyable.

I also want to thank my family, for all the support they gave me while I was away from home in order to make this project possible; and my friends who always desired me the best.

Abstract

The aim of this thesis is the implementation of a successful equilibrium control in a humanoid robot, so it is able to stabilize itself and avoid the fall when a disturbance is applied to it, using the Zero Moment Point as the reference. For this purpose, robot TEO from the RoboticsLab at University Carlos III of Madrid is used. In the case of sensory perception, the chosen sensor is an Inertial Measurement Unit, which provides useful data about linear accelerations around the center of mass. The system also includes a low pass filter, seeking to reduce the noise in the IMU measurements, and a PID controller, in order to compare the actual state with the desired one and offer an output. The equilibrium control consists of two different strategies: Ankle Strategy and Hip Strategy. One or the other will come into action depending on the magnitude of the disturbance.

In order to achieve the goal of this thesis, knowledge in programming, robotics and physics is needed, as well as a clear understanding of key concepts on equilibrium.

Keywords: balance, stability, equilibrium, humanoid robot, zero moment point, inertial perception, push recovery, ankle strategy, hip strategy.

Chapter 1

Introduction

Our environment is entirely adapted to humans, as we are the ones living in it. That is the main reason why recently the design of humanoid robots is experiencing major development. A robot similar to a human in size, shape and movements has the great advantage of not having to adapt the environment to it. In addition, a humanoid robot is able to perform more human tasks than the rest and perform them better, as it is mechanically similar. However, building a humanoid robot is complex. The situations that a human would overcome instinctively need to be taken into account in the design of this kind of robots, such as the ability of keeping balance, which is the main focus of this thesis. When it comes to equilibrium, we must take into account any possible influences in the system. It can suffer two types of disturbances (Petrovic, Jovanovic, & Potkonjak, 2014): the ones caused by the system itself (e.g. when moving arms) and the external ones (e.g. unpredicted disturbances). In contrast to the vast majority of the papers, which perform stabilization controls only in the sagittal plane, we are implementing a two dimensional control. We could say that it is in fact a 3D control, as the robot moves through the three dimensions, but we won't be controlling its position in the z-axis or its movement in the axial/transversal plane, so it is more accurate to name it as a 2D control.

For this project we will be using the robot's ankles and hip. TEO has hip joints, which connect the leg to the trunk, and waist joints, placed above the hip and used for moving the trunk in the sagittal and axial planes. The sagittal waist joint is more useful for our purpose rather than the actual hip joint. This project is bio-inspired (Martínez, 2012), and to avoid confusion I must clarify that during the whole thesis, for convenience with the human model, we will be refering as the "hip" to what, actually, is the "waist". So, even though we will be using TEO's waist joints, we will talk about "Hip Strategy", for instance.

After taking a look at the motivation and objectives of the project, this thesis is organized as follows. First, a brief introduction to robotics history and development until these days. Then, literature about bipedal equilibrium, including the basics, models and equilibrium control systems. After that, the system's architecture is presented: the proposed control system, robot platform, sensor, controller and communication platform. The next chapter exposes the general characteristics of the software used. Then we have information about the assembly of the sensor in the robot, the development of the project and the experiments performed with their results. Finally, the thesis ends up with some conclusions and proposed future projects.

1.1 Motivation of the thesis

At the RoboticsLab from University Carlos III of Madrid we are currently developing a human-size humanoid called TEO, which will perform different kind of tasks as it is intended to be inserted as assistance robot in the future. This robot is being used for research in many different areas such as manipulation tasks, computer vision, non-grasping manipulation using force/torque sensors, bipedal locomotion or the main point of this project: whole-body balance.

Developing a two-dimensional equilibrium control for a human-size robot is an intriguing project with multiple implications in other works and future applications. It presents an interesting challenge in the field of robotics, and specifically in humanoid robotics. The ability to stabilize by itself from a disturbance is a very useful advance for any humanoid robot and a key factor in the successful creation of a functional human-like robot. The development of robots able to work in the human environment is the current point of research in robotics.

1.2 Objectives

The final objective of the project is to design and develop a successful two-dimensional equilibrium control for humanoid robot TEO from University Carlos III of Madrid. However, this involves other objectives fulfilled during the process:

- Sensory perception using an Inertial Measurement Unit.
- Correct computation of the Zero Moment Point.
- Design of satisfactory PID controllers.
- Communication with the robot's devices via YARP.
- Practical implementation of the equilibrium control in the physical robot.
- Push-recovery experiments with the humanoid robot as a way of checking the successful performance the designed control.

Chapter 2

Brief introduction to robotics

The word *robot* appeared for the first time in 1920 in the science fiction play called *R.U.R. (Rossumovi Univerzální Roboti)*, written by Czechoslovakian Karel Čapek, and translated to English years later (Čapek, 1923) (figure 2.1). This term comes from the Czech word *robota*, which means servitude or forced labor, and it is used in the novel for referring to these machines whose only function is to be a workforce in the service of humans. *Robot* is synonym of serf. *Robotics* is a science fiction term that appeared in this novel and from which the research regarding this topic began.

Figure 2.1: *Rossum's Universal Robots, by Karel Čapek*

Nevertheless, Jacques de Vaucanson is considered to be the creator of the first robot, *The Flute Player* (figure 2.2), in 1738. It was a real size figure of a shepherd which could play 12 different songs with a flute.

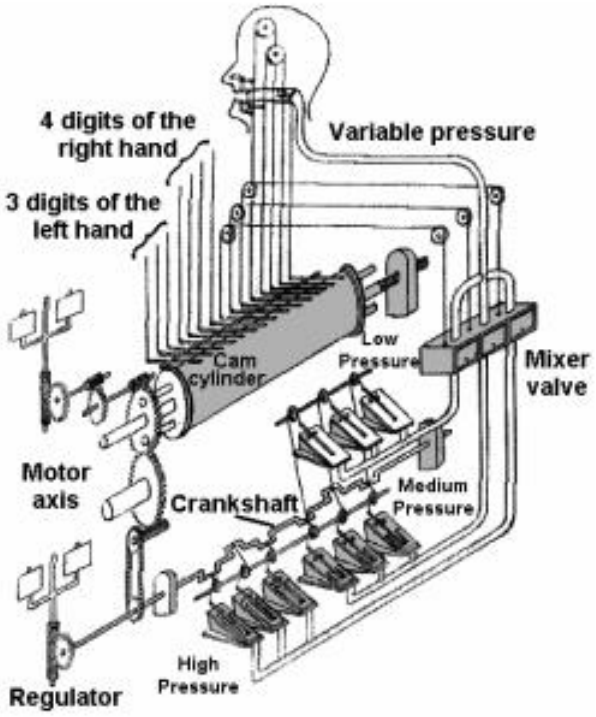

Figure 2.2: *The Flute Player's mechanical details*

In 1927 appears the first film with robots: *Metropolis* (Lang, 1927) (figure 2.3). However, robotics didn't attract attention until the most known publication in the robotics world: *I, Robot* (Asimov, 1950) (figure 2.4). This book included the famous three laws of robotics:

- A robot can't hurt a human being or stay inactive during damage to him.
- The robot must obey the human being except if it contradicts the first law.
- The robot must protect his existence unless it is in conflict with the previous laws.

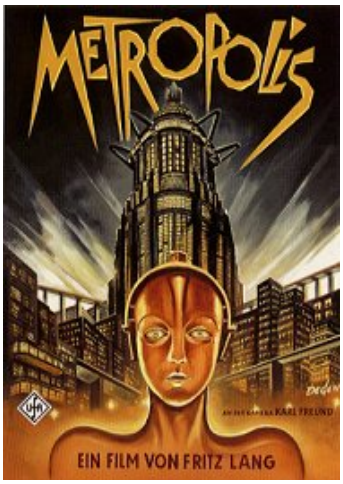

Figure 2.3: *Metropolis, by Fritz Lang*

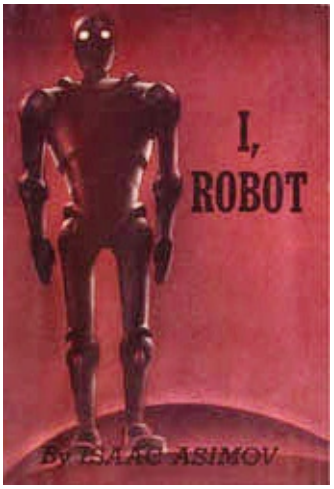

Figure 2.4: *I, Robot, by Isaac Asimov*

In 1948, Grey Walter had already built the first mobile robots, known as *Bristol's tortoises* (figure 2.5). Walter used valves, light sensors, contact detectors, two driving wheels, a photo-tube as an eye, etc.

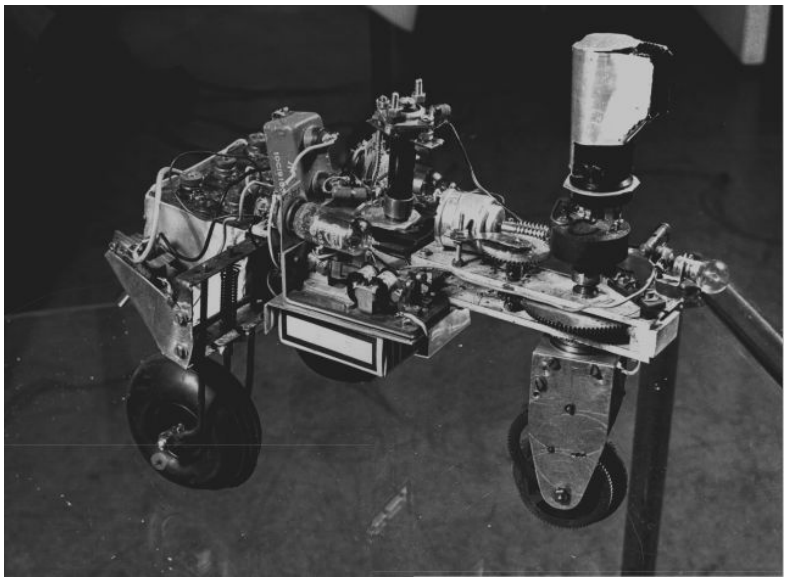

Figure 2.5: *Grey Walter's tortoise*

Until the 50's, robots were servomechanisms teleoperated by a human being. They had no self-control system. But in 1954, the first programmable robot was patented by engineer George Devol. Two years later, Devol and businessman Josef Engelberger created the first company dedicated to robotics: Unimation (Universal Automation). In 1961, the first Unimate's robot was installed in a General Motors' plant (figure 2.6), used for manipulation of material in a foundry machine.

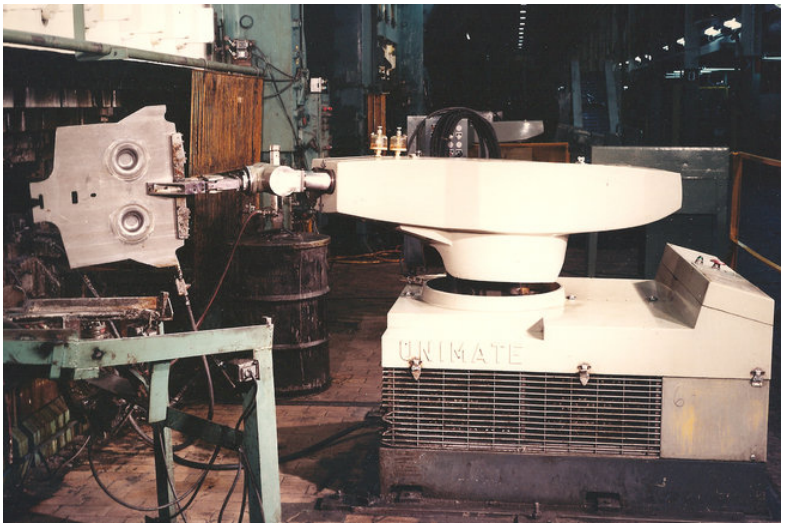

Figure 2.6: *George Devol's robotic arm*

In the 70's, several remarkable robots appeared. In 1970, *Shakey* robot was created by scientists from Stanford University and its task was to process an image and plan the right movement. In 1971, the *Mars 2* probe (also called *Marsnik 2)* was launched by the Soviet Union and it became the first human object to reach Mars. In 1978 appears the robot PUMA (*Programmable Universal Machine for Assembly*), created by Unimation and General Motors, which consisted on a programmable manipulator arm (figure 2.7).

Figure 2.7: *Unimation's PUMA*

Since the creation of the first robotic company in 1956, robotics has advanced quickly, starting with simple robotic manipulator arms and arriving to space mobile robots, such as the first rover (space exploration vehicle) in the surface of Mars: the *Sojourner* (figure 2.8). It was part of the *Mars Pathfinder* mission, launched in 1966.

Figure 2.8: *The Sojourner studying the rock Yogi after leaving the lander*

2.1 Humanoid robots

A humanoid robot is a robot with anthropomorphic form designed to imitate the movements of a human being. In general, humanoid robots have a chest, a head, two arms and two legs, although sometimes only a part of the body is modelled, for instance from the waist to the head. Some robots can have a head designed to replicate human facial features, such as the eyes and the mouth. Androids are humanoid robots built to look aesthetically like humans.

While industrial or mobile robots need to adapt to the environment, humanoid robots have the advantage of being capable of working directly in the same environment as humans. The applications of humanoid robots are very varied: personal assistance (ARMAR III, HRP-2), education (ASIMO), entertainment (Hitachi, Hubo, New ASIMO), security (Honda P3, Guardrobo), medicine, search and rescue, etc.

The first design of a robot humanoid was done by Leonardo da Vinci in 1495. Nowadays we know it as *Leonard's Robot* (figure 2.9) and it had the appearance of a mechanical knight.

Figure 2.9: *Leonard's Robot, built in 2007 by Mario Taddei*

The first humanoid robot created was the *Kaufmann Trumpeter* (figure 2.10), made by Friedrich Kaufmann in 1810 in Dresden (Germany). It consisted in an automatic trumpeter of the size of a human with head, chest and two legs, but only one arm.

Figure 2.10: *Kaufmann Trumpeter*

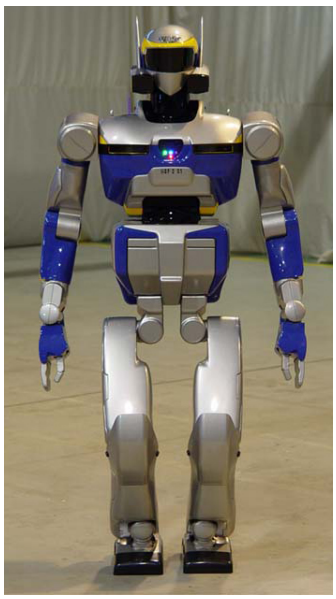

Figure 2.11: *Kawada HRP-2 (MSTC)*

Since then, lots of different humanoid robots have been created. Recently, one of the most advanced and known humanoids in the world was built: robot ASIMO, made by Honda in 2000. Two years later, at the Manufacturing Science and Technology Center (MSTC) of Tokyo, the *Kawada HRP-2* (figure 2.11) was created, which is considered the most advanced humanoid robot.

Figure 2.12: *Robot ASIMO going upstairs*

Nowadays, research in humanoids is focused mostly in bipedal locomotion. That is to say, the robot to be capable of walking correctly on two legs keeping balance. Once this is achieved, more complex objectives can be proposed, such as running, going upstairs (figure 2.4) or downstairs.

Chapter 3

Bidepal equilibrium

This section refers to the ability of a humanoid body to keep balance. Stabilization is an important matter if the aim of the robot is to be able to walk on two legs or perform any task without falling. There are several significant points about this matter which must be discussed.

3.1 Basics of bipedal equilibrium

3.1.1 Center of Mass

A concept to take into account is the Center of Mass (CoM). The CoM of a discrete or continuous system is the geometric point that dynamically behaves as if the resultant of the external forces were applied to it. In the case of a single rigid body, the Center of Mass is fixed in relation to the body. It may be located outside the physical body, as is sometimes the case for hollow or open-shaped objects, such as a horseshoe. In the case of a distribution of separate bodies, such as the planets of the Solar System, the Center of Mass may not correspond to the position of any individual member of the system.

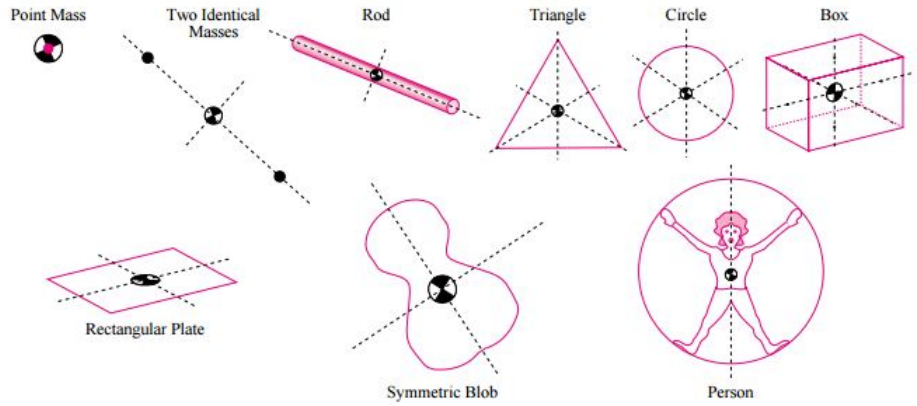

Figure 3.1: *Examples of Center of Mass (Ruina & Pratap, 2015)*

3.1.2 Center of Gravity

In physics, a Center of Gravity of a material body is a point that may be used for a summary description of gravitational interactions. It is a point in a body around which the resultant torque due to gravity forces vanishes. In a uniform gravitational field, the Center of Mass coincides with the Center of Gravity. For practical purposes, this coincidence is met with acceptable accuracy for almost all bodies that are on the Earth's surface, even for a locomotive or a large building, since the decrease in gravitational intensity is very small to the full extent of these bodies.

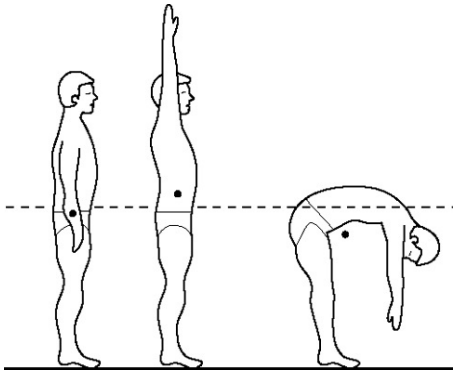

Figure 3.2: *Center of Gravity of people*

3.1.3 Center of Pressure

In biomechanics, Center of Pressure (CoP) is the term given to the point of application of the ground reaction force vector. The ground reaction force vector represents the sum of all forces acting between a physical object and its supporting surface. Analysis of the Center of Pressure is common in studies on human postural control and gait.

The Center of Pressure is not a static outcome measure. For instance, during human walking, the Center of Pressure is near the heel at the time of heelstrike and moves anteriorly throughout the step, being located near the toes at toe-off. This concept is important when working with Force/Torque sensors, which gathers data about the surface pressures for calculating the CoP.

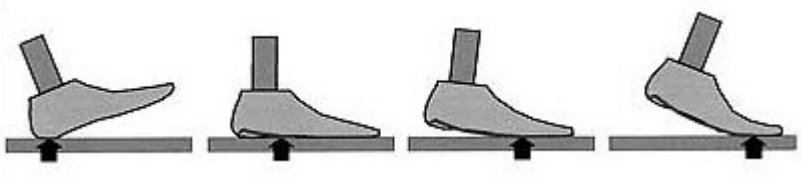

Figure 3.3: *Center of Pressure in a foot's sole while walking*

3.1.4 Centroidal Moment Pivot

The Centroidal Moment Pivot is the point where the ground reaction force would have to act to keep the horizontal component of the whole-body angular momentum constant. The CMP is equal to the CoP in the case of zero moment about the Center of Mass (figure 3.4). However, humanoids have many internal joints, particularly the torso and arms, that allow them to apply a torque about the CoM. For a non-zero centroidal moment, the CMP moves beyond the edge of the support polygon (figure 3.5).

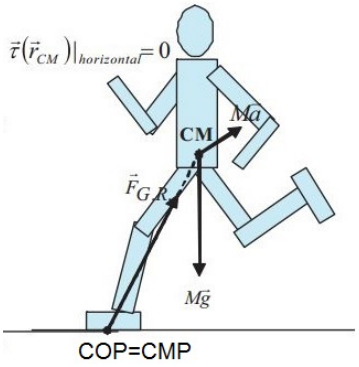

Figure 3.4: *CMP with zero moment about the CoM*

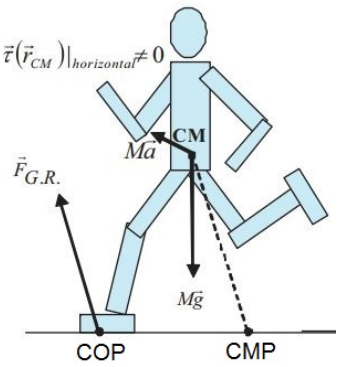

Figure 3.5: *CMP with non-zero moment about the CoM*

3.1.5 Zero Moment Point

One of the most important concepts in stabilization is the so called Zero Moment Point (ZMP). It is the point respect to which the dynamic reaction force in the contact between foot and floor does not produce any moment in horizontal direction, i.e., the point in which the total of the horizontal inertia and gravitational forces is equal to zero. This concept assumes that the contact area is flat and has enough friction to avoid the foot from sliding. The ZMP is one indicator of the stability of the robot (figure 3.6).

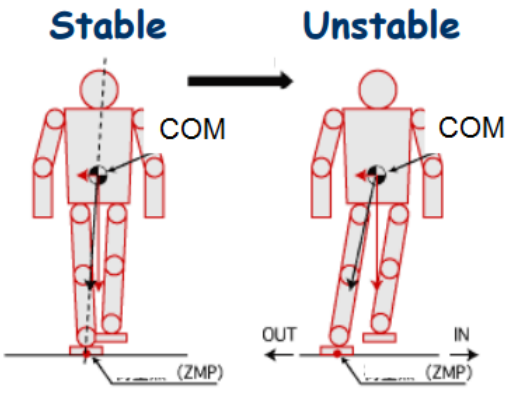

Figure 3.6: *Stability according to the ZMP*

3.2 Models

3.2.1 Linear Inverted Pendulum Model

The LIPM (Kajita & Tani, 1991) is a simple way of modelling a robot which has multiple uses in robotics. Modern research in postural control and bipedal locomotion has been heavily influenced by this model.

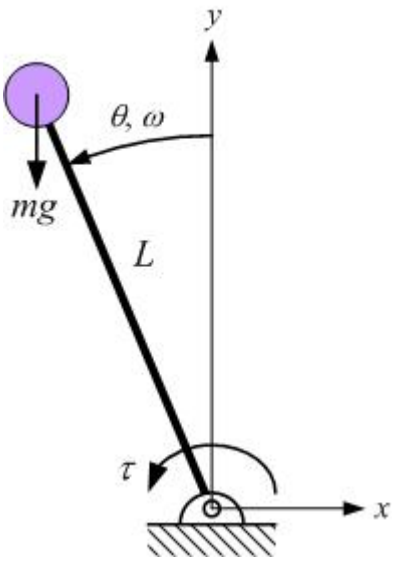

Figure 3.7: *Linear Inverted Pendulum Model*

In figure 3.7, m is the total mass of the system concentrated in the CoM, linked through a massless link to a pivot point, which represents the robot's feet. L is the distance from the robot's feet to its CoM. These dynamics correspond to a two-dimensional system representing humanoid motion, but it can be extended to three dimensions. When considering ankle torques and the constraints on the location of the ZMP it has also been referred to as the cart-table model.

3.2.2 Cart-table model

A publication made by researchers of the National Institute of Advanced Industrial Science and Technology of Japan (Kajita et al., 2003) exposes the mathematical reasoning for obtaining the Zero Moment Point in a 3D lineal inverted pendulum model. Authors conclude that the equations needed in order to calculate coordinates x and y of the ZMP (coordinate z is zero due to the fact that the point must be in the floor) are the following:

$$p_x = x - \frac{z_c}{g}\ddot{x} \tag{3.1}$$

$$p_y = y - \frac{z_c}{g}\ddot{y} \tag{3.2}$$

Where:

p_x, p_y = coordinates x and y of the ZMP.

x, y, z_c = distance to the Center of Mass in the three axes.

$\ddot{x}$, $\ddot{y}$ = acceleration in axes x and y.

g = acceleration of the gravity.

Cart-table is a model which directly corresponds to these equations. It depicts a running cart of mass m on a pedestal table whose mass is negligible (figure 3.8). Two sets of this model are needed for the motion of x and y. When the cart has a determined acceleration, the table maintains itself during a specific time, in which the ZMP exists inside the foot of the table. It is possible to obtain the coordinates of the ZMP by using this model and its equations in the xz and the yz planes.

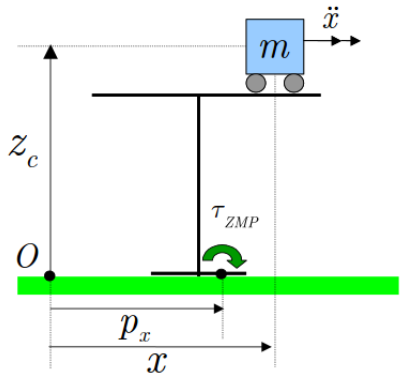

Figure 3.8: *Cart-table model*

On the other side, other authors (Kajita & Espiau, 2008) add a complement to the equation. According to them, when the cart acceleration is too large, the computed ZMP goes outside of the support polygon. This happens since the previous equations do not consider the support polygon and the unilateral constraint. In other words, it assumes that the foot is glued to the floor, which is not viable because the foot should rise so the robot can walk. In order to realize the correct calculation, we have to take into account the acceleration in the z axis, apart from the gravity.

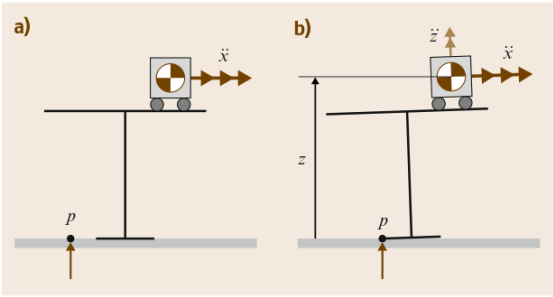

Figure 3.9: *Computed ZMP (a) fictitious case (b) correct result*

In figure 3.9, case (a) shows the ZMP without taking into account that acceleration and case (b) taking it into account.

With this change, the equation suffers a light modification. Simply, acceleration in the z axis ($\ddot{z}$) has been added.

$$p_x = x - \frac{z_c}{g + \ddot{z}}\ddot{x} \tag{3.3}$$

$$p_y = y - \frac{z_c}{g + \ddot{z}}\ddot{y} \tag{3.4}$$

This is important because the ZMP appearing outside of the support polygon implies that the robot could have the foot not in total contact with the floor and the motion when walking can be not as planned. When the ZMP is inside the support polygon, the total contact foot-floor is guaranteed.

Although it may seem that there is no acceleration in the z axis, or that is almost negligible, it is advisable to add it in the equation because it doesn't affect in case of being zero, but improves the calculation of the ZMP in case of existing.

3.3 Equilibrium Control

The aim of this control is to develop a reflex dedicated to prevent the fall of the robot, using information provided by its sensors. The principle is to compensate the tilt of the robot when a disturbance is applied. If the robot can be stabilized by itself, it will use its joints and motors to prevent the fall.

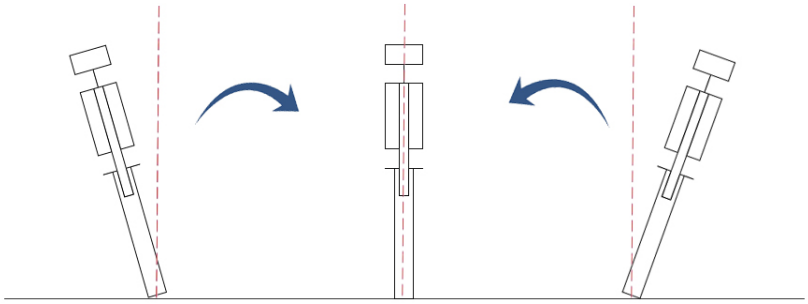

Figure 3.10: *Side view of robot falling*

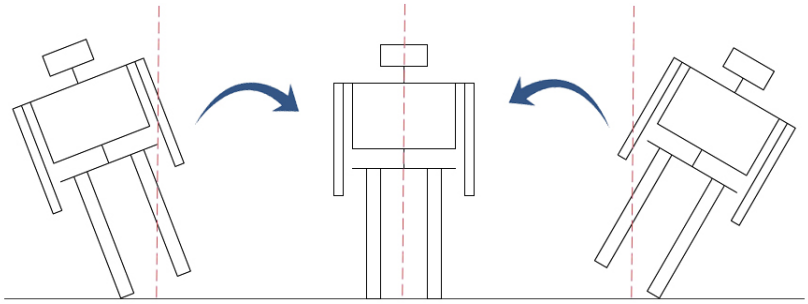

Figure 3.11: *Frontal view of robot falling*

Researchers have been studying balance in bipedal robots since years ago, and this topic has suffered major development in recent years. Some authors have studied the forces acting on humanoid robots (Sardain & Bessonnet, 2004) and others have even developed balance strategies . The stability control can be designed in so many different ways. Benjamin Stephens (Carnegie Mellon University) explores different models and strategies used for humanoid balance (Stephens, 2011). He presents three basic strategies:

- CoP Balancing (Ankle Strategy)
- CMP Balancing (Hip Strategy)
- Stepping (Change-of-support Strategy)

Generally, these strategies can be employed sequentially from top to bottom, advancing to the next if the current strategy is inadequate. For small disturbances, simply behaving like an inverted pendulum and applying a compensating torque at the ankle can be enough. As the disturbance increases, however, more of the body has to be used. Bending the hips or swinging the arms creates an additional restoring torque. Finally, if the disturbance is too large, the only way to stop from falling is to take a step.

Stephens chooses the CoP and the CMP as the reference points for the equilibrium control. However, other points can be considered for being the reference, such as the Zero Moment Point (Vadakkepat & Goswami, 2008).

In this project, we have chosen to use the ZMP as the reference stability point. Despite the fact of using another point for the control, we based our strategies in Stephen's proposals.

3.3.1 Ankle Strategy

This is the simplest balancing strategy, which uses ankle torque to apply a restoring force, while the other joints are fixed. It consists in an ankle reaction torque proportional to the ZMP.

What Stephens calls "decision surfaces" are functions of reference points which predict whether or not a fall is inevitable using a specific control strategy. These are useful for determining when can we stick to the lower strategy or we need to jump in to more complex ones to make the recovery. The decision surface proposed to check if the Ankle Strategy is viable is the following:

$$\delta^- < \frac{\dot{x}_{CoM}}{\omega} + x_{CoM} < \delta^+ \tag{3.5}$$

Where:

δ^- = back edge of support polygon

δ^+ = front edge of support polygon

$\dot{x}_{CoM}$ = CoM velocity

x_{CoM} = CoM position

$\omega = \sqrt{\frac{g}{Zcom}}$

Constants g and $Zcom$ refer to the gravitational acceleration and the distance from the origin of coordinates to the CoM in the z-axis, respectively.

If the state of the humanoid is outside of this decision surface, then ankle torque alone cannot restore balance and either a different balance strategy is needed or a step should be initiated to prevent falling.

3.3.2 Hip Strategy

When the disturbance is so strong that the state of the humanoid is outside the decision surface in (3.5), the Ankle Strategy is not enough for restoring balance, so the Hip Strategy comes into action. This strategy consists in increasing the restoring torque by creating a moment about the CoM using the hip torques. This is to say, not only the ankles will react creating a restoring torque, but also the hip.

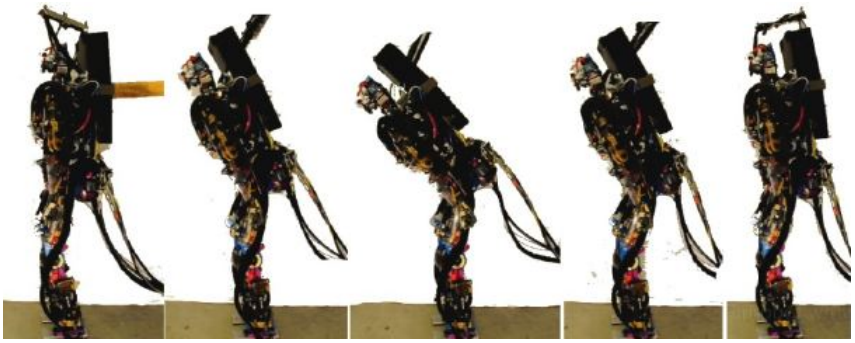

Figure 3.12: *Push recovery using Hip Strategy*

In the case of the Hip Strategy, the decision surface is:

$$\delta^- - \frac{\tau_{max}}{mg}(e^{\omega T_{max}} - 1)^2 < x_0 + \frac{\dot{x}_0}{\omega} < \delta^+ + \frac{\tau_{max}}{mg}(e^{\omega T_{max}} - 1)^2 \quad (3.6)$$

Where:

δ^- = back edge of support polygon

δ^+ = front edge of support polygon

τ_{max} = maximum torque

m = mass

g = gravity

$\omega = \sqrt{\frac{g}{Zcom}}$

T_{max} = maximum time

x_0 = initial position

$\dot{x}_0$ = initial velocity

If the system is within this stability region, then the system can be stabilized using the Hip Strategy. However, if the system is outside, it will have to take a step to avoid falling over.

3.3.3 Stepping

For even larger disturbances, no amount of body torquing will result in balance recovery. In this case, it is necessary to take a step. As said previously, this condition can be checked with the decision surface (3.6). Taking a step moves the support region to either the area between the rear leg heel and the forward leg toe (double support) or the area under the new stance foot (single support). The stronger the disturbance is, the larger the step needs to be.

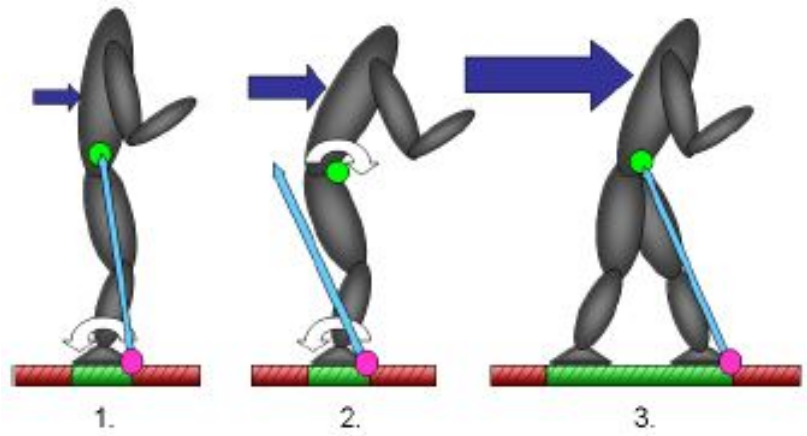

Figure 3.13: *Push-recovery by stepping*

Stephens also implements a stepping strategy (Stephens & Atkeson, 2010b). Other authors dig deeper into this matter introducing more complex controls, which involve topics such as additional step to place feet side by side, knee bending (Assman, 2012), walking on slippery ground (Hofmann, 2006), and much more.

Chapter 4

System Architecture

4.1 Control System

As discussed in previous chapters, other authors use the Center of Pressure (CoP balancing), Centroidal Moment Pivot (CMP balancing) or other relevant points in order to perform control strategies. However, in this project we use Kajita's model and we aim to stabilize the Zero Moment Point, so it could be called ZMP balancing.

Basically, what we try to accomplish is to generate a reaction in the robot's joints according to the inputs we obtain. The inputs will be the linear accelerations obtained by inertial perception. Then, using the cart-table model, the ZMP is computed. A PID controller offers an output according to the difference between this ZMP and the desired setpoint. Finally, this output is sent as a velocity command to the robot joints' motors.

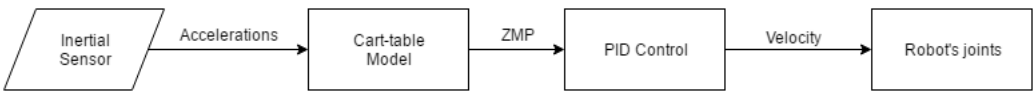

Figure 4.1: *Block diagram of the control system*

4.2 Robot TEO

The humanoid robot used for this project is the *Task Environment Operator* (TEO), developed by the RoboticsLab at the University Carlos III of Madrid (Monje et al., 2011). This robot weights approximately 60 kg and it is 1'70 meters tall.

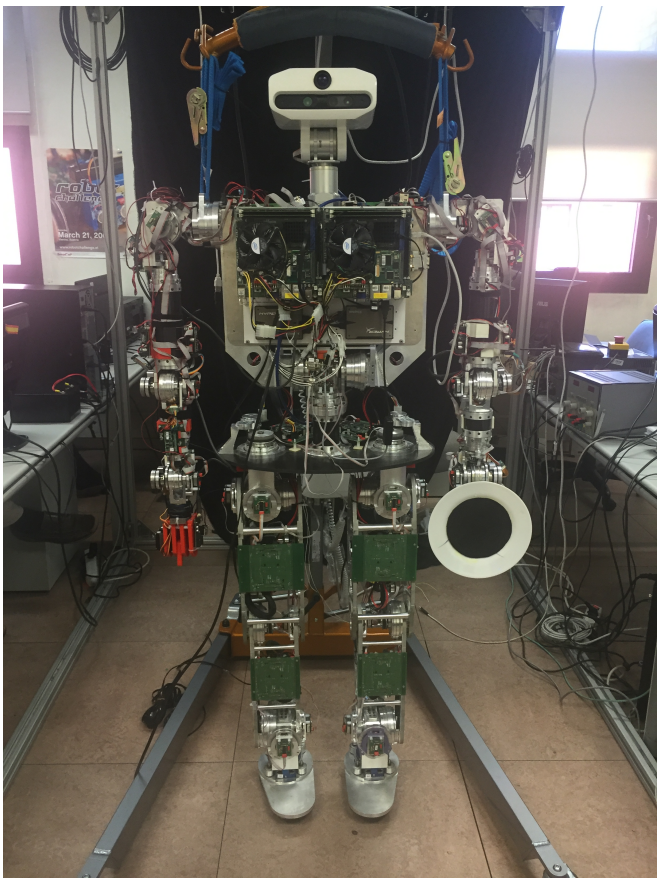

Figure 4.2: *Robot TEO (UC3M)*

Humanoid robot TEO is a mechanical system of 28 degrees of freedom, which are distributed as shown in figures 4.3 and 4.5 and table 4.2. Its lengths are shown in figure 4.4 and table 4.1.

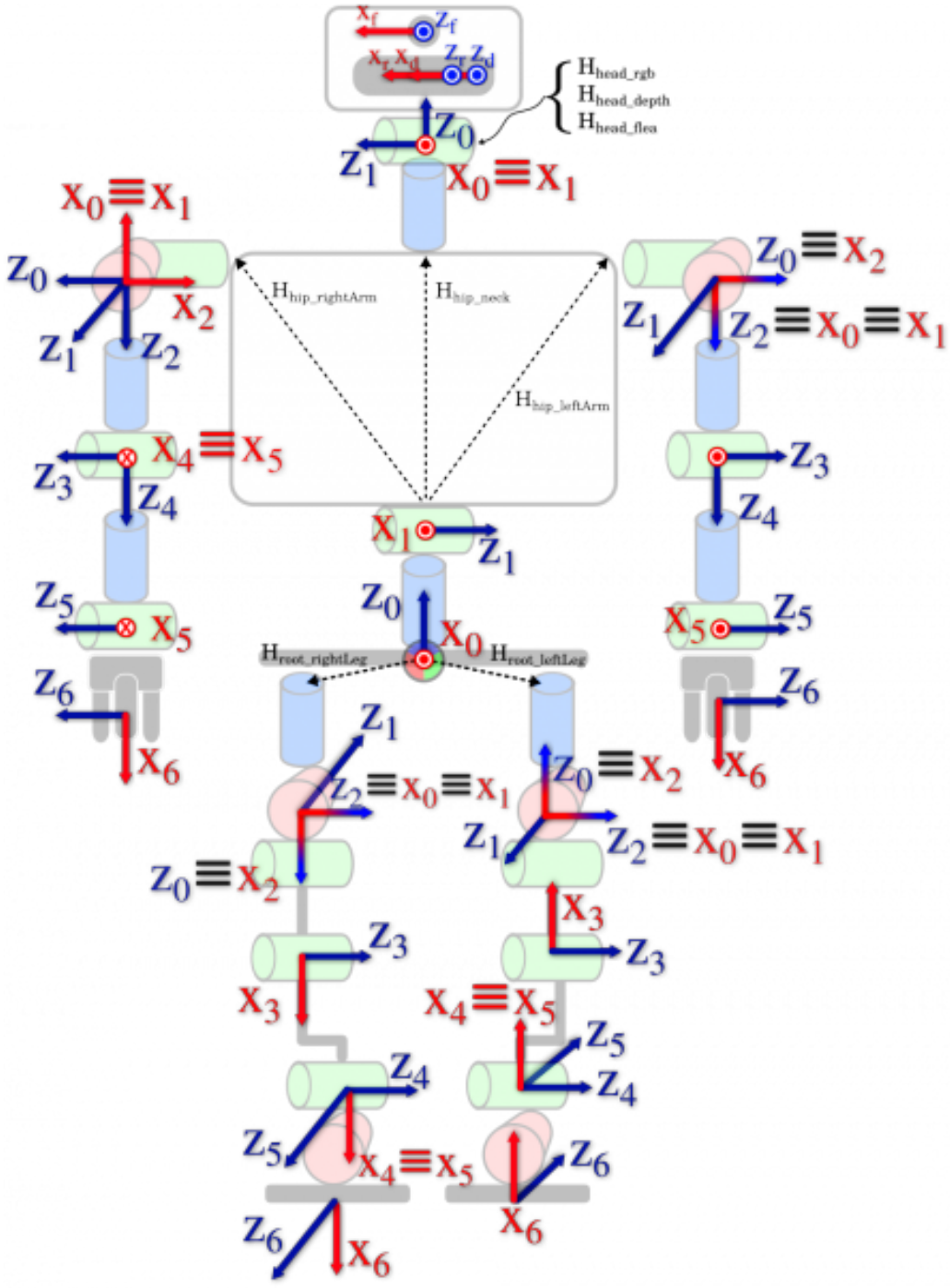

Figure 4.3: *TEO's degrees of freedom*

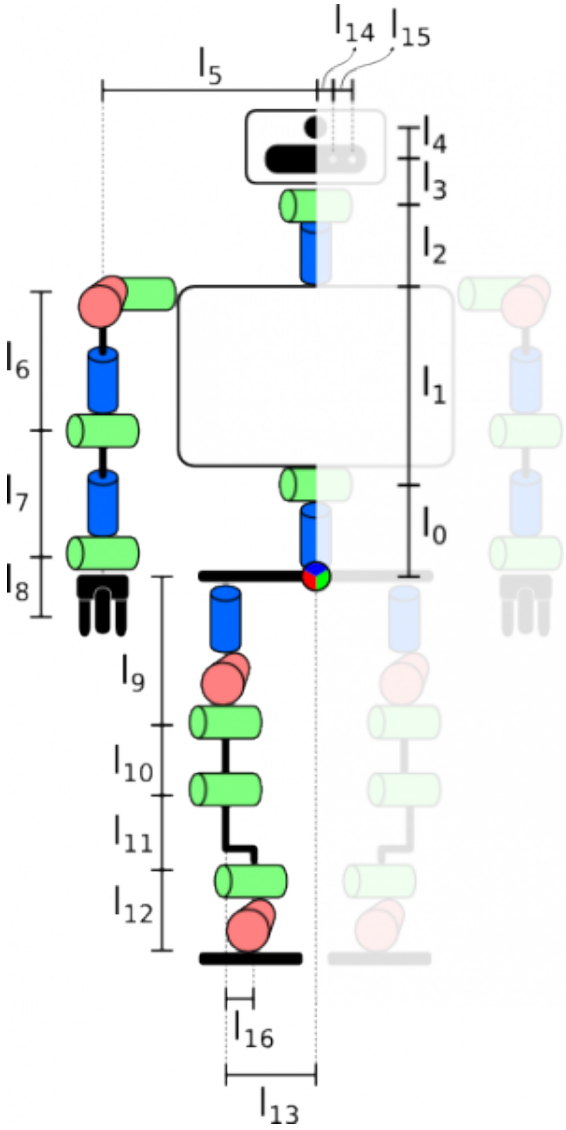

Figure 4.4: *TEO lengths*

	L1	L2	L3	L4	L5	L6	L7	L8	L9
mm	330	33.22	300	124	146	305	338	337	210

Table 4.1: *Measurements of figure 4.4*

PART	JOINT	DEGREES OF FREEDOM	
Leg	Hip	3	6
	Ankle	2	
	Knee	1	
Arm	Shoulder	2	6
	Elbow	2	
	Wrist	2	
Trunk	Waist		2
Head	Neck		2
TOTAL			6x2 + 6x2 + 2 + 2 = **28**

Table 4.2: *Distribution of DOF in robot TEO*

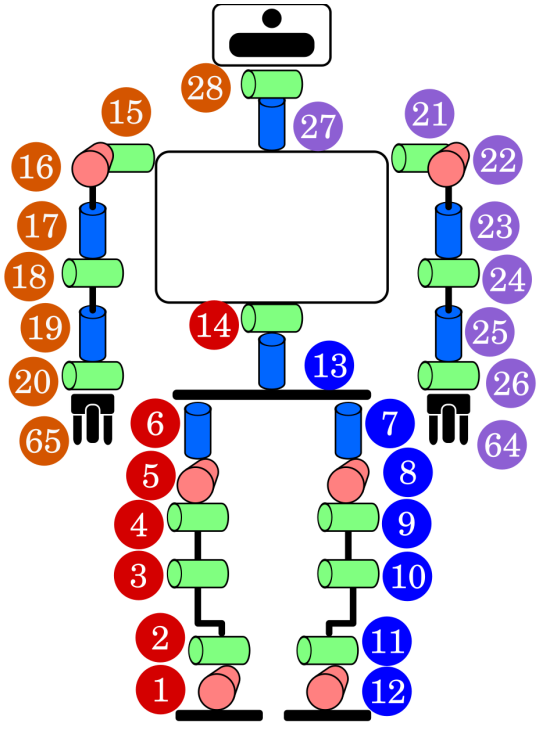

Figure 4.5: *TEO joints*

4.3 Inertial Sensor

As previously mentioned, linear accelerations of the system are obtained by inertial perception. For this purpose, an Inertial Measurement Unit (IMU) is used, which is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers, gyroscopes and magnetometers.

IMUs are the main component of inertial navigation systems used in aircraft, spacecraft, watercraft, drones, UAV and guided missiles among others. In this capacity, the data collected from the IMU's sensors allows a computer to track a craft's position, using a method known as dead reckoning. An Inertial Measurement Unit is composed of different components:

- Accelerometer: Device that measures proper acceleration (*g-force*) in applications of tilt measurements, as well as dynamic accelerations resultant of movements or collisions. Proper acceleration is not the same as coordinate acceleration (rate of change of velocity).

- Gyroscope: Device that measures the change in orientation in x, y and z axes.

- Magnetometer: Measurement instrument used to measure the magnetization of a magnetic material or to measure the strength and, in some cases, the direction of the magnetic field at a point in space. It can be used to know the deviation respect to the Earth's magnetic north in degrees, being 0 the north and 180 the south.

Some IMUs can also include other components as a barometer, which offers data of temperature, pressure and altitude.

4.3.1 Xsens MTi-28A53G35

The MTi is a miniature size and low weight 3DOF Attitude and Heading Reference System (AHRS). The MTi contains accelerometers, gyroscopes and magnetometers in 3D, and as such is an Inertial Measurement Unit (IMU) as well. Its internal low-power signal processor provides real-time and drift-free 3D orientation as well as calibrated 3D acceleration, 3D rate of turn and 3D earth-magnetic field data. Specifically, the sensor used for this project is the MTi-28A53G35 (figure 4.6).

Figure 4.6: *MTi-28A53G35, from Xsens*

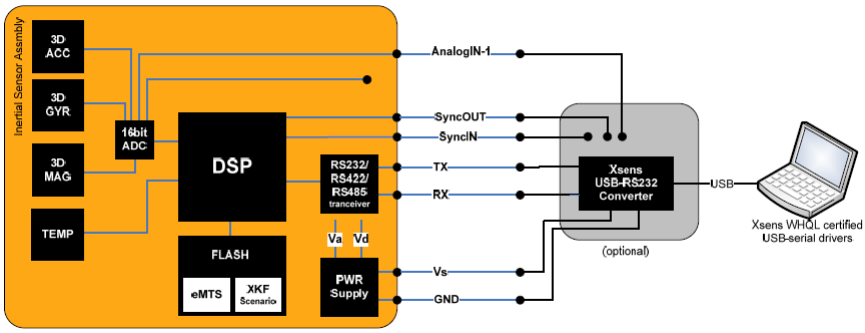

Figure 4.7: *Modules inside the MTi sensor*

The MTi is an excellent measurement unit for stabilization and control of cameras, robots, vehicles and other (un)manned equipment. It has a compact and robust design, and is easy to integrate in any system or application. Its

weight and consumption are low. It uses gyroscopes, accelerometers and magnetometers in order to determine the orientation in a range of 360º, using gravity and the Earth's magnetic field as references. The sophisticated Xsens sensor fusion algorithm copes with temporary magnetic disturbances and short-term accelerations, resulting in a reliable real-time orientation estimate. Additionally, the MTi SDK (Software Development Kit) incorporates a magnetic field mapping routine to correct for hard and soft iron effects, and provides interfaces at multiple levels: intuitive Windows GUI software, API binary libraries (Windows, Linux) as well as supplying source code implementing the MTi binary communication protocol for easy integration on any platform (XsensTechnologies, 2011).

	Mti-28A♯♯G♯♯
Communication interface	Serial digital (RS-232)
Additional interfaces	SyncIn SyncOut Analog In
Operating voltage	4.5 - 30 V
Power consumption (AHRS/3D orientation mode)	350 mW
Temperature Operating Range	-20°C - 55°C
Specified performance Operating Range	0°C - 55°C
Outline Dimensions	58 x 58 x 22 mm (W x L x H)
Weight	50 g

Table 4.3: *Characteristics of sensor MTi-28A53G35*

4.3.2 IMU sensor integration

Sensor assembly

Where to place the IMU in the robot is a topic whose conclusion depends on several factors. First, the closer the sensor is to the COM, the better. In the cart-table model (section 3.2.2), for calculating the ZMP we need to know the accelerations of the cart, which is the COM in the physical robot. Thus, the best place to mount the IMU is the closest you can to the COM. However, it must be in a place where doesn't conflict with the body of the robot when it moves. For instance, it can't block the rotation of a body part. Also, it must not conflict with the wires of the robot. In addition, the closer it is to where it will be plugged, the better, so the length of the wire is reduced as much as possible. And it has to be somewhere easily accessible, in case you need to remove it for testing or any reason. Finally, for obtaining the accelerations it is preferable to place the IMU straight and not inclined. In our case the decision is easy. TEO has an accessible free space with no conflicts under the plate where the battery is placed (figures 4.8 and 4.9), and it is around the area where the COM is supposed to be.

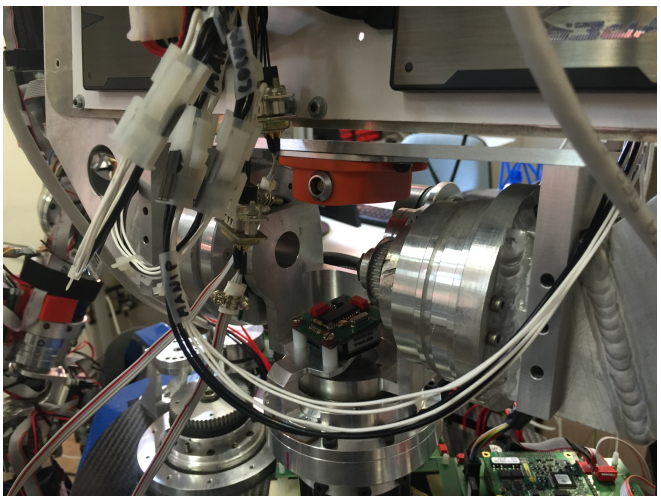

Figure 4.8: *IMU mounted in TEO (1)*

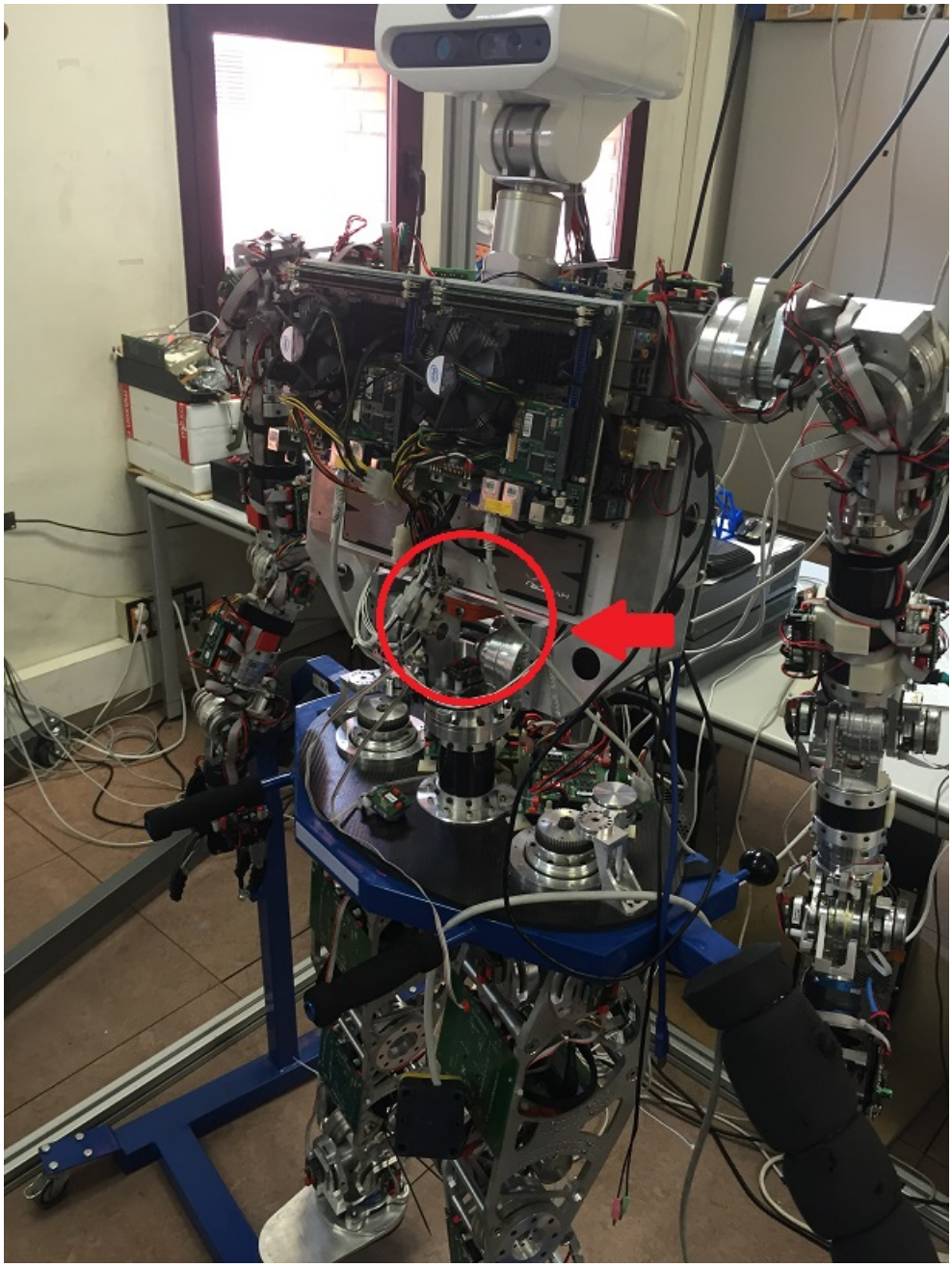

Figure 4.9: *IMU mounted in TEO (2)*

Sensor disassembly

There is something important to say regarding the decoupling of the sensor. If, for any reason, it is necessary to remove the sensor, it must be done by unscrewing the whole piece under the battery (figure 4.10). In order to do this, the six screws specified in figure 4.11 must be unscrewed.

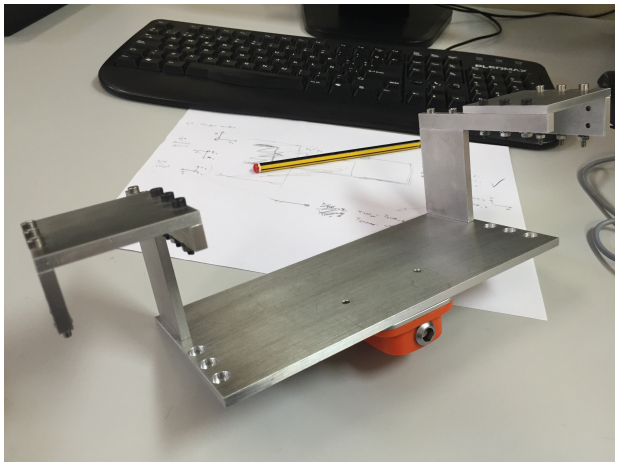

Figure 4.10: *Whole piece to be disassemblied*

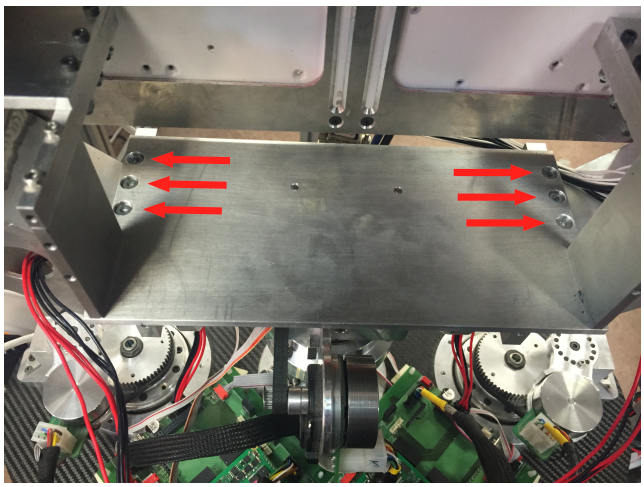

Figure 4.11: *Screws to be unscrewed*

Connection cable

This specific MTi sensor uses a RS-232 cable (CA-USB2) like the one seen in figure 4.12, which includes a USB converter.

Figure 4.12: *RS-232 cable*

The problem with this USB-serial data and power cable was its length. It was 2.5 meters long, which is too much. The solution was to shorten the cable to a reasonable length for its connection to the robot's CPU and buying a long USB cable extender so the sensor could be connected to an external computer for testing, as can be seen in figure 4.13.

Figure 4.13: *New RS-232 cable and long USB*

TEO has three different CPUs:

- Vision (Head)
- Manipulation (Arms and neck)
- Locomotion (Legs and trunk)

The IMU sensor will be connected to the locomotion CPU as seen in figure 4.14.

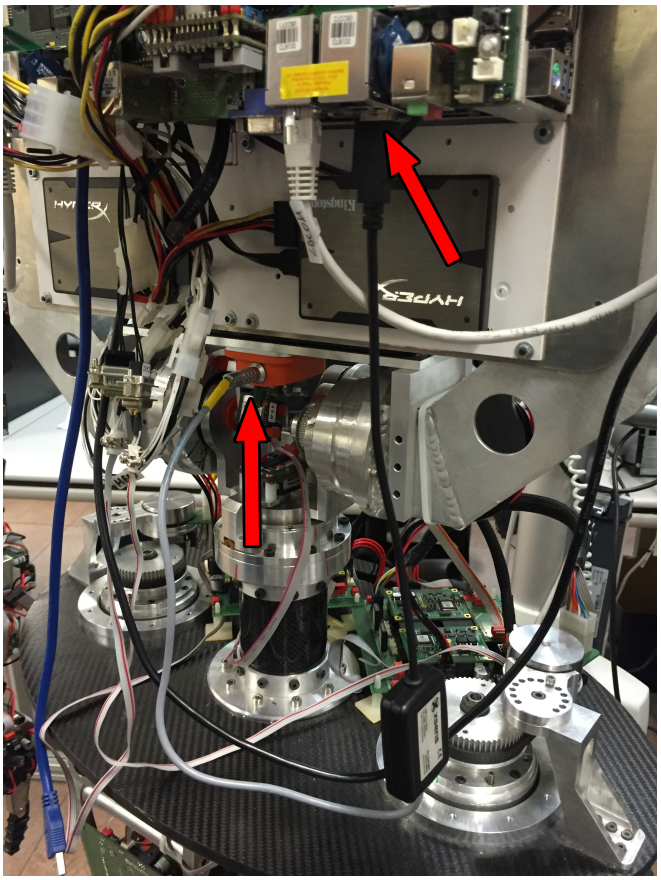

Figure 4.14: *MTi sensor connected to locomotion CPU*

4.4 PID control

In order to control the robot's actuators depending on the sensor's measurements, a PID controller is used in the project. A proportional–integral–derivative controller (PID controller) is a control loop feedback mechanism commonly used in industrial control systems. A PID controller continuously calculates an error value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the error over time by adjustment of a control variable, such as the position of a control valve, a damper, or the power supplied to a heating element, to a new value determined by a weighted sum:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{4.1}$$

where K_p, K_i, K_d denote the coefficients for the proportional, integral, and derivative terms, respectively (sometimes denoted P, I, and D).

- P accounts for present values of the error.
- I accounts for past values of the error.
- D accounts for possible future values of the error, based on its current rate of change.

By tuning the three parameters of the model, a PID controller can deal with specific process requirements. The response of the controller can be described in terms of its responsiveness to an error, the degree to which the system overshoots a setpoint, and the degree of any system oscillation.

Some applications may require using only one or two terms to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions.

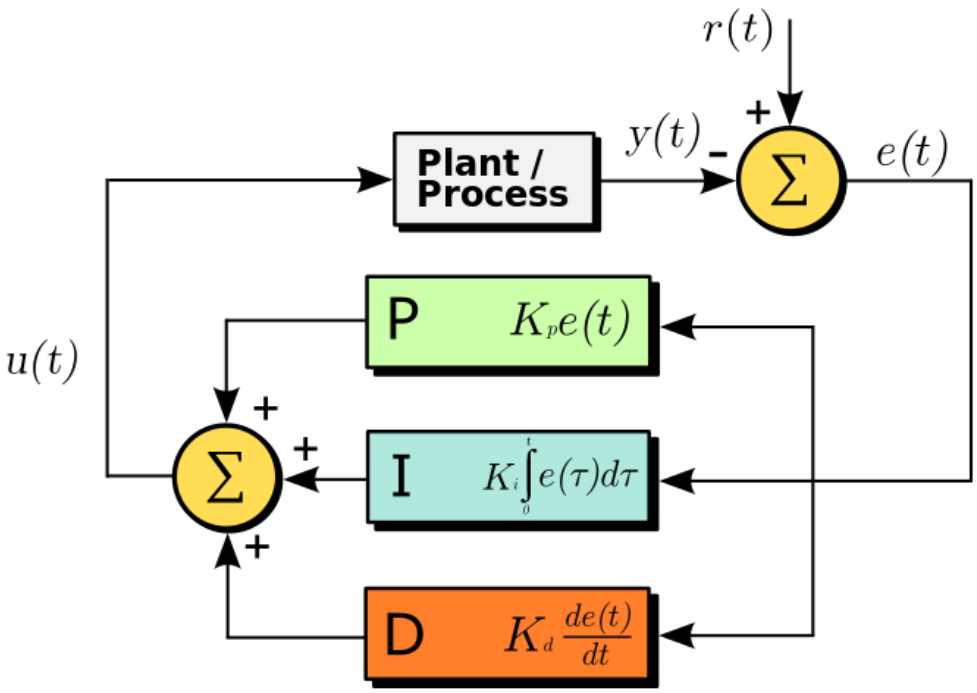

Figure 4.15: *Block diagram of a PID controller in a feedback loop*

4.4.1 Proportional action

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant. The proportional term is given by:

$$P_{out} = K_p e(t) \tag{4.2}$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances.

4.4.2 Integral action

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The integral term is given by:

$$I_{out} = K_i \int_0^t e(\tau)d\tau \tag{4.3}$$

The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value.

4.4.3 Derivative action

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain Kd. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, Kd. The derivative term is given by:

$$D_{out} = K_d \frac{de(t)}{dt} \tag{4.4}$$

Derivative action predicts system behavior and thus improves settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low pass filtering for the derivative term, to limit the high frequency gain and noise.

4.5 Tools

The project has been developed in C++. It was chosen mainly because the robot is being developed using this programming language, and also because of the knowledge I have about it over others. The only exception is the use of external Python applications. While the code is completely written in C++, Python is used only for obtaining more visual results when performing tests. This language was chosen with this purpose because the Python library *matplotlib* is very useful for plotting graphs.

4.5.1 C++

C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation (*cplusplus*, 2016).

It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, servers (e.g. e-commerce, web search or SQL servers), and performance-critical applications (e.g. telephone switches or space probes). C++ is a compiled language, with implementations of it available on many platforms and provided by various organizations, including the FSF, LLVM, Microsoft, Intel and IBM.

Bjarne Stroustrup, a Danish computer scientist, began his work on C++'s predecessor "C with Classes" in 1979 at Bell Labs. In 1983, C++ appears as an extension of the C language as Stroustrup wanted an efficient and flexible language similar to C, which also provided high-level features for program organization. From the object oriented languages point of view, C++ is a hybrid language.

4.5.2 Python

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. (*Python*, 2016)

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI (Centrum Wiskunde & Informatica) in the Netherlands.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code. (Hunter, 2007)

4.5.3 iCub

Figure 4.16: *iCub*

The iCub robot (figure 4.16) is a 104 cm tall humanoid with a weight of 22 kg, developed at the Italian Institute of Technology inside the RobotCub international consortium, in which several European universities are involved (Metta, Sandini, Vernon, Natale, & Nori, 2008). This robot has 53 degrees of freedom. Some of the iCub's sensors are cameras, microphones, inertial sensors, force/-torque sensors, position sensors or tactile sensors. Its Open Source software was developed on top of YARP.

The importance of iCub's software remains in the fact that it has already implemented the access to a MTx inertial sensor. TEO re-uses this software in order to access to its MTi sensor. For this reason, iCub's software is installed in TEO's locomotion CPU.

4.6 Communication

The communication platform used for communicating with the different devices included in TEO is called YARP. (*YARP: Yet Another Robot Platform*, 2016)

YARP is free and open software. Along with many other benefits, the *Free Software* social contract can speed software development for small communities with idiosyncratic requirements. It is a set of libraries, protocols, and tools to keep modules and devices cleanly decoupled. It is reluctant middleware, with no desire or expectation to be in control of your system.

YARP is an attempt to make robot software that is more stable and long-lasting, without compromising the ability to constantly change our sensors, actuators, processors, and networks. It helps organize communication between sensors, processors, and actuators so that loose coupling is encouraged, making gradual system evolution much easier. The YARP model of communication is transport-neutral, so that data flow is decoupled from the details of the underlying networks and protocols in use (allowing several to be used simultaneously). YARP uses a methodology for interfacing with devices that again encourages loose coupling and can make changes in devices less disruptive.

YARP is written by and for researchers in robotics, particularly humanoid robotics. Running decent visual, auditory, and tactile perception while performing elaborate motor control in real-time requires a lot of computation. The easiest and most scalable way to do this right now is to have a cluster of computers. Every year what one machine can do grows, but so do our demands. YARP is a set of tools useful for meeting our computational needs for controlling various humanoid robots.

The components of YARP can be broken down into:

- *libYARP_OS*: Interfacing with the operating system(s) to support easy streaming of data across many threads across many machines. YARP is written to be OS neutral, and has been used on Linux, Microsoft Windows, Mac OSX and Solaris. YARP uses the open-source ACE (ADAPTIVE Communication Environment) library, which is portable across a very broad range of environments, and YARP inherits that portability. YARP is written almost entirely in C++.
- *libYARP_sig*: Performing common signal processing tasks (visual, auditory) in an open manner easily interfaced with other commonly used libraries, for example OpenCV.
- *libYARP_dev*: Interfacing with common devices used in robotics: framegrabbers, digital cameras, motor control boards, etc.

Chapter 5

Development

5.1 Data processing

The first step is to obtain data from the MTi sensor. Thanks to the iCub software, we are able to access to the YARP port transmitting the IMU measurements, called */inertial*. Through this port, the information received consists of twelve values in the following order:

- Euler angles [3]: deg
- Linear acceleration [3]: m/s^2
- Angular speed [3]: deg/s
- Magnetic field [3]: arbitrary units

In order to calculate the Zero Moment Point afterwards, and according to equations (3.3) and (3.4), we will need to get the CoM accelerations. Consequently, The relevant values are the fourth, fifth and sixth.

When obtaining data from the inertial sensor, measurements have noise. In order to reduce this noise, a low pass filter has been implemented. Specifically, a filter computing the average of the last n samples. This average is what is used for calculating the ZMP afterwards. Thanks to this, the Zero Moment Point is more accurate. For each iteration, the new measurement is saved and the oldest is deleted (figure 5.1). This way, we have always the same number of measurements to compute the average and these are constantly updated. With this process, the increase in computation time and memory usage is negligible.

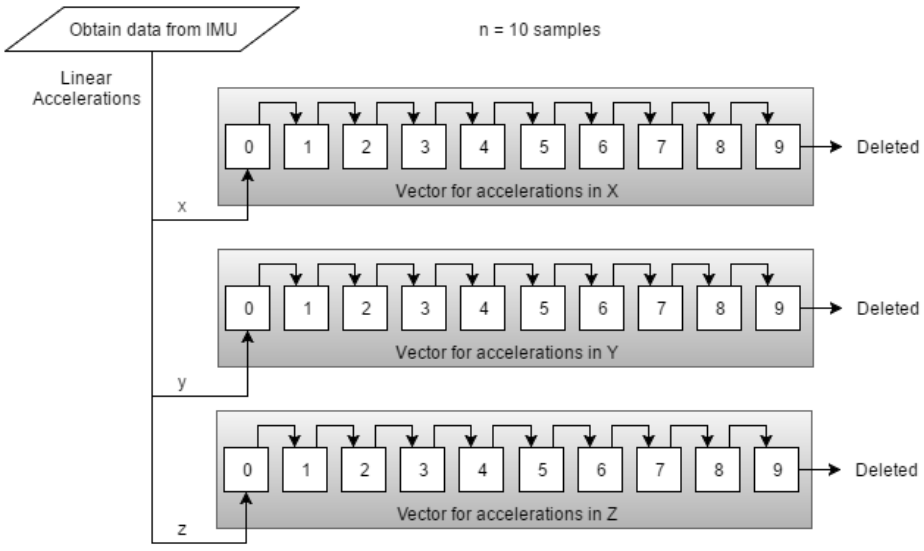

Figure 5.1: *Vector of measurements*

5.2 Conversion to robot's coordinate system

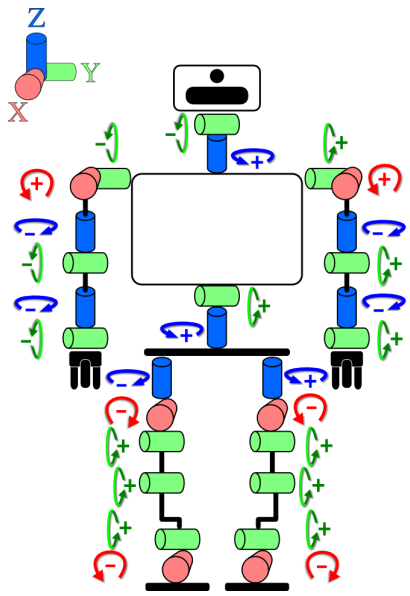

Figure 5.2: *TEO's coordinate system and joint directions*

Due to the fact that the sensor has been placed upside down (see figure 4.8), the values must be converted to the robot's coordinate system (figure 5.2). We must take into account that the sensor has two possible configurations for its coordinate system (XsensTechnologies, 2009): the one usually set by default (figure 5.3) and the one corresponding to the North-East-Down (NED) convention (figure 5.4).

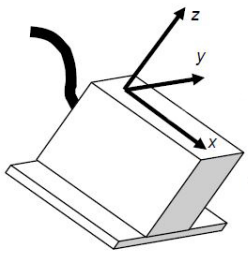

Figure 5.3: *Default coordinate system*

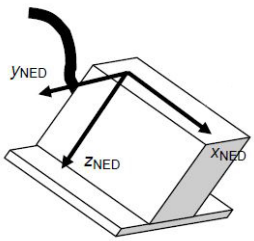

Figure 5.4: *NED coordinate system*

By experimentation, it was found out that the NED was the system set. Then, taking into account that coordinate system and the position in which the sensor was placed (figure 4.8), the conversion from the sensor's coordinate system to the robot's coordinate system is the following:

$$x_{robot} = x_{sensor}$$

$$y_{robot} = -y_{sensor}$$

$$z_{robot} = z_{sensor}$$

5.3 ZMP computation

According to the cart-table model explained in section 3.2.2, we can compute the Zero Moment Point by using equations (3.3) and (3.4). For using these equations we need the x, y and z accelerations obtained before (after the conversion to the robot's coordinate system) and the distances from the origin of coordinates to the CoM in each axis. The CoM is located as seen in figure 5.5. The distance to this CoM in x and y is zero, and for the distance in z we can check figure 4.4 and table 4.1.

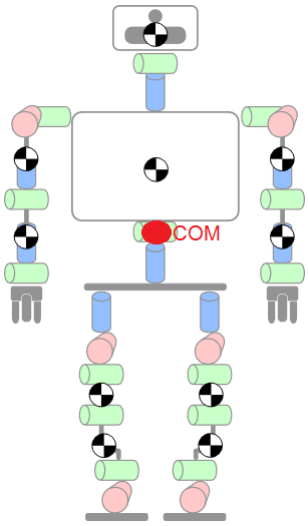

Figure 5.5: *TEO's Center of Mass*

The result of the equations will be a point with x and y coordinates: $ZMP(p_x, p_y)$. There is no z coordinate because the Zero Moment Point is a projection on the floor, so z is always zero. It is important to highlight that the acceleration of gravity should not be added to the total z-axis acceleration ($\ddot{z}$). This is because the inertial sensor already takes into account this acceleration.

5.4 PID control

We design a PID controller so it gives an output in relation to the coordinate of the ZMP, which is the input. We need to design two PID controllers: one for the ankles and one for the hip. The parameters needed for the constructor in the PID library used are:

- Loop interval time in seconds.
- Maximum output value.
- Minimum output value.
- Proportional term (Kp).
- Derivative term (Kd).
- Integral term (Ki).

The loop interval time is the time that passes since one iteration starts until so does the next one. A 10 ms value was chosen in principle.

The maximum and minimum output values are the velocity limits of the joint motor. TEO's motors are configured so that the maximum velocity is 10 deg/s. By performing velocity tests, we find that a maximum of 10 deg/s is enough for our purpose. The minimum velocity for the controller is the same as the maximum but with a negative value, which means that it will go in the opposite direction.

Finally, the control parameters (Kp, Kd, Ki) need to be set. The drivers of the motors in the robot include their own PID, so, for a first attempt, the same parameters are chosen. The only change needed is related to the loop interval time. As the driver works 10 times faster than our controller, the PID parameters will be increased tenfold. This parameters will be adjusted during the experimental tests.

The parameters of the PID controller in the ankle motor's driver are:

- Kp = 0.79987
- Ki = 0.015869
- Kd = 0

The parameters of the PID controller in the hip motor's driver are:

- Kp = 0.14999
- Ki = 0.00054932
- Kd = 0

As you can see, the derivative term is equal to zero, so firstly we are implementing a PI controller.

The used library includes a function in charge of performing the PID control and offering an output. In order to do this, the actual value and the setpoint must be entered. The actual value is the previously calculated x or y coordinate of the ZMP (depending on if we are working in the sagittal or the frontal plane), and the setpoint is its desired value.

The setpoint is set by the user and depends on the desired configuration of the robot. The ideal value of the ZMP for the robot standing still is (0, 0). However, if TEO is not in the "home position", we can take the first computed ZMP as reference, and use this value as setpoint. Nevertheless, this variable can be changed by the user at will.

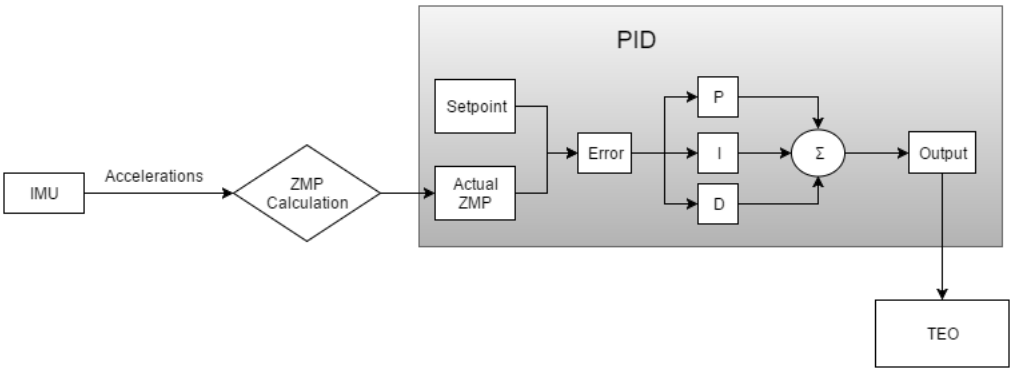

Figure 5.6: *PID block diagram*

5.5 Joint reaction

The last step only consists in sending the PID output to the joint motor as a velocity in deg/s. The result must meet that the more difference between the actual ZMP and the desired ZMP, the higher reaction velocity is implemented. In order to send a motion command to a robot's joint, we must connect to the specific device, and then send the command to the specific motor. According to figure 5.7, for moving the ankles, for instance, we must connect to devices */teo/rightLeg* and */teo/leftLeg*, and send the motion command to motors number 4 or 5 (depending on the plane).

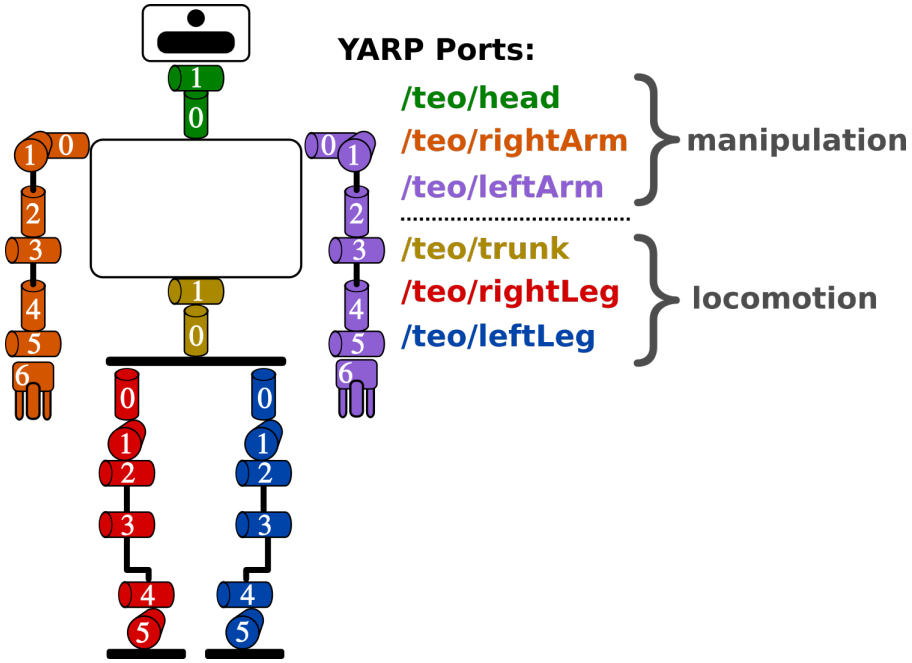

Figure 5.7: *TEO YARP ports*

The motion command may involve a position or a velocity. In this case, the command used transmits a velocity in deg/s by a single value which also establishes the direction of the velocity depending on if it is positive or negative. As it can be seen in figure 5.2, a positive value for the ankle joint means that the robot would try to lean forwards, and a negative value would be the opposite.

5.6 Equilibrium control

Once everything seemed to work fine, the final step was trying to put all together in order to implement the different strategies and, finally, achieve a valid full push-recovery control.

5.6.1 Sagittal Ankle Strategy

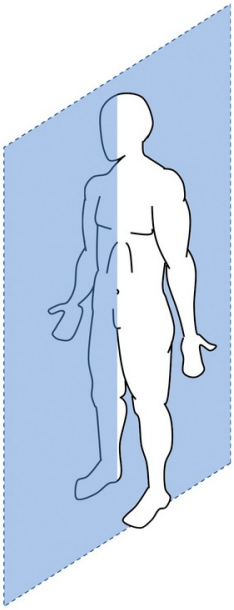

Figure 5.8: *Sagittal Plane*

For this strategy we are using joints 2 and 11, according to figure 4.5. The ankle motors will receive a velocity command depending on the PID control output, which, in turn, depends on the IMU measurements. Combining the different tests was not difficult. It was only a matter of establishing the velocity output obtained in the PID control, as the velocity input to send to the motor. Also, as we are only implementing the Ankle Strategy, determining the strategy is not necessary.

An important change made in order to correctly react to a disturbance was the sign of the PID, so it corresponds with the correct motion desired according to figure 5.2.

5.6.2 Sagittal Hip Strategy

For this strategy we are using joints 2, 11 and 14, according to figure 4.5. The first step for combining ankle and hip strategies is to implement the determining of strategy. This must be done after the ZMP is computed in order to check if we need an Ankle Strategy or a Hip Strategy, and then control the system with the correspondent controllers.

Strategy determining

In order to correctly choose which is the appropriate strategy for the stability control, we use equation 3.5. If the state of the robot meets the conditions of this equation, then the robot can stabilise itself only with the Ankle Strategy. If the conditions are not fulfilled, then it is necessary to jump to the Hip Strategy.

The edges of the support polygon (δ^- and δ^+) in the x-axis are the edges of TEO's feet. The length of the feet is 24 cm, but as the origin of coordinates is in the middle point, then the edges will be at $\pm$ 12 cm ($\pm$ 0.12 m). The value of ω can be easily calculated.

$$\omega = \sqrt{\frac{g}{Zcom}} \tag{5.1}$$

Where:

g = gravitational acceleration = 9.81 m/s^2

$Zcom$ = distance to the CoM in the z-axis = 1.036602 m

This values are constant.

The Zero Moment Point is almost equal to the projection of the COM on the floor, so we can use the x coordinate of the computed ZMP as the position of the COM (X_{CoM}).

Finally, the last unknown value is the linear velocity of the COM ($\dot{X}_{CoM}$). We get the linear accelerations from the sensor, so obtaining the linear velocity is just a matter of multiplying the acceleration to the time.

Now that we are getting all needed information, we can use the equation for determining a strategy. In case we were also implementing the final strategy (stepping), we would have to take into account another equation to check if the Hip Strategy is enough for avoiding the fall or we need a step.

Hip control

Once we have set the parameters for choosing the correct strategy, now it is time to develop the control of the Hip Strategy. For this purpose, we will be adding the hip's counteraction to disturbances to the previously implemented ankles' motion.

The procedure for TEO to stabilize itself depends on the magnitude of the disturbance. When a disturbance is applied to the robot, the system will determine if the Ankle Strategy is enough or not, depending on equation 3.5. Then we have two possibilities:

- Ankle Strategy: If the Ankle Strategy is enough, then the ankles will act counteracting the leaning of the robot until the stability position has been reached, as seen previously when testing this process.

- Hip Strategy: If the Ankle Strategy is not enough, then the hip joins the ankles in trying to stabilize the robot's position. This motion will be carried out until it is determined that the Hip Strategy is no longer needed. Then, the hip will return to its home position and the ankles will be the only ones in charge to stabilize the robot, as in the Ankle Strategy.

For the hip's PID controller, we have to take into account that the error is not very important. This is because the ankles are the ones finally stabilizing the robot after the hip's action. We could say that the hip is responsible to avoiding the fall, and the ankles to recover the stable position. What is really important for the hip's PID is the reaction time. For a first attempt, we start testing with the same PID parameters used for the Ankle Strategy, but with 15 samples for the filter instead of 30.

5.6.3 Frontal Strategy

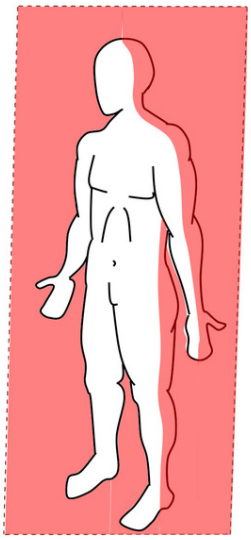

Figure 5.9: *Frontal Plane*

Performing push-recovery tasks in the frontal plane is different from what was done before. When working in the sagittal plane, leaning back and forth, the robot acts as a linear inverted pendulum with single support, as seen in figure 3.7. However, in the frontal plane, the robot has double support, so the control is different.

In this case, we have to solve the stability problem with a different approach. We will be using the frontal motors of the ankles and the hip (joints 1, 5, 8 and 12 according to figure 4.5). The motion of the robot's legs seeks to keep the torso straight after a disturbance and recovering the initial position, as seen in figure 5.10.

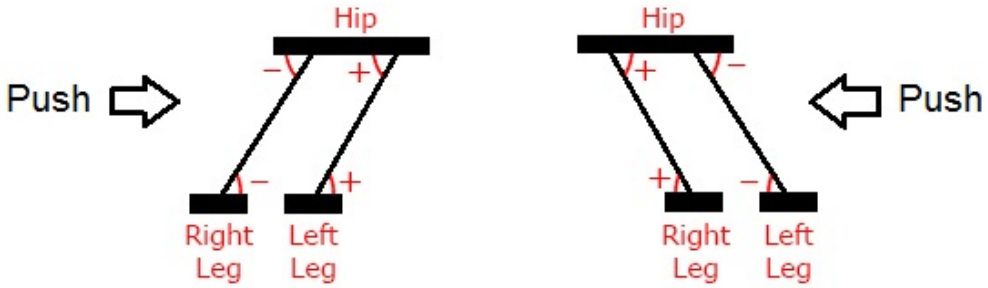

Figure 5.10: *Leg's motion in Frontal Strategy*

TEO's hip only has two joints: the sagittal joint (used previously for the Sagittal Hip Strategy) and the joint to rotate on itself around the z-axis. TEO doesn't include a joint for moving on the frontal plane, and that is why the hip can't be used as in the Sagittal Hip Strategy.

5.6.4 2D Equilibrium Control

Now that the strategies have been implemented for two dimensions, it is time to put all together and try it at the same time. We will be using all the joints together: joints 1, 2, 5, 8, 11, 12 and 14 (figure 4.5). It has been programmed so there are two threads running in parallel: one for the sagittal control and one for the frontal control.

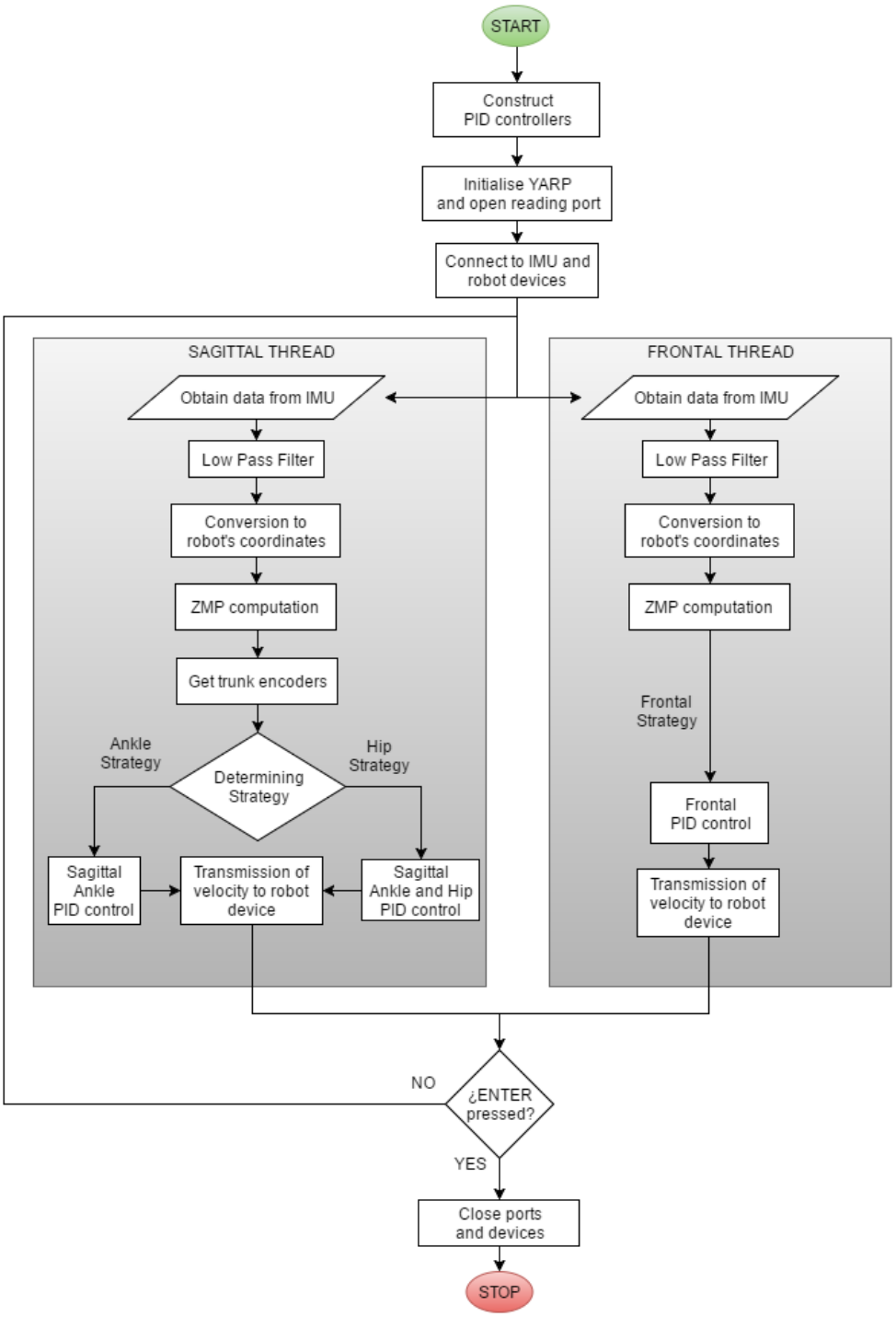

Figure 5.11: *System's flowchart*

Chapter 6

Experiments and Results

6.1 Data processing

Data from the inertial sensor is easily obtained. As it can be seen in figure 6.1, the MTi sensor provides information about euler angles, linear accelerations, angular speed and magnetic field. For this project, the only relevant data are the linear accelerations, which will be used for calculating the ZMP afterwards.

```
Starting /home/teo/Documents/inertialSensor/build/inertialSensor...
yarp: Port /inertial:i active at tcp://163.117.150.179:10004
yarp: Receiving input from /inertial to /inertial:i using tcp

Obtaining data from sensor...

Angle X: -179.782 deg
Angle Y: -0.0593454 deg
Angle Z: 40.5635 deg
Acceleration X: 0.0255452 m/s²
Acceleration Y: -0.0532082 m/s²
Acceleration Z: -9.79509 m/s²
Angular speed X: -0.346812 deg
Angular speed Y: 0.409501 deg
Angular speed Z: 0.499629 deg
Magnetic field X: 0.28206
Magnetic field Y: 0.247834
Magnetic field Z: 0.733974
```

Figure 6.1: *Data obtained from the IMU sensor*

With the purpose of testing the filter, experiments were performed with different number of samples. Figures 6.2, 6.3 and 6.4 show the accelerations in the x-axis as they are obtained from the sensor (red) and after computing the average (blue). As it can be seen, the more samples the filter uses, the more it smooths the average. Sometimes maybe too much. In addition, the more samples it uses, the more offset appears, so it will take more time to the control to react to this acceleration.

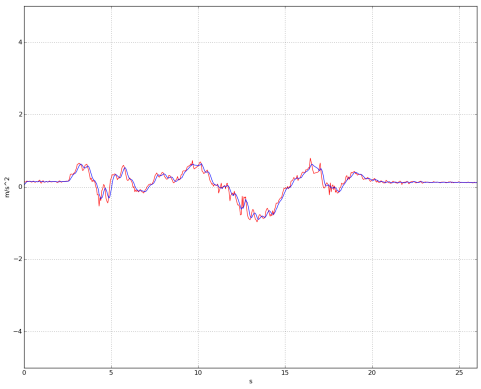

Figure 6.2: *LP filter with 5 samples*

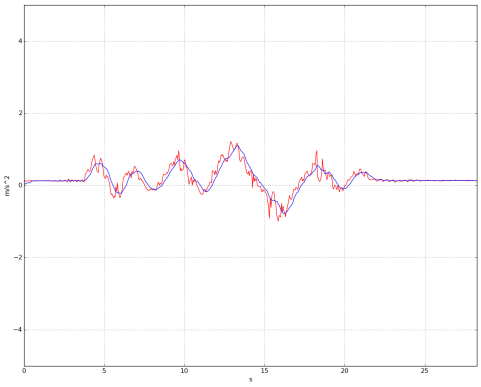

Figure 6.3: *LP filter with 10 samples*

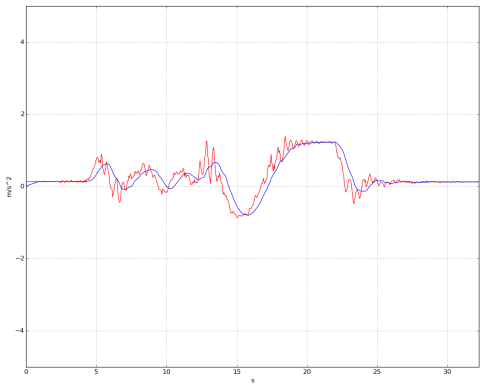

Figure 6.4: *LP filter with 20 samples*

For a deeper analysis, tests were performed in order to check the error in the ZMP computed after using the filter, with the robot standing still and not moving.

Samples	ZMPx error (cm)
1	$\pm$ 0.239
5	$\pm$ 0.098
10	$\pm$ 0.061
20	$\pm$ 0.059

Table 6.1: *ZMP error for different LP filters*

For a first attempt, pending the final experimental tests, the number of samples chosen for the filter is 10.

6.2 ZMP computation

The aim of this test was to correctly calculate the Zero-Moment Point. The resultant point can be plotted online in order to check if the results are reasonable. In figures 6.5 to 6.9 you can see a plot representing both feet of the robot (based on real measurements) and the resultant ZMP for five different situations depending on its leaning.

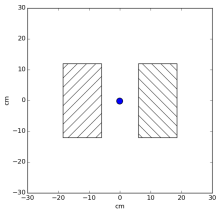

Figure 6.5: *ZMP when standing still*

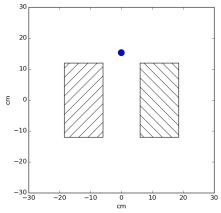

Figure 6.6: *ZMP when leaning forwards*

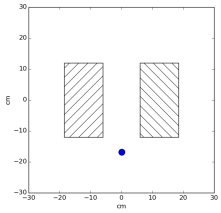

Figure 6.7: *ZMP when leaning rearwards*

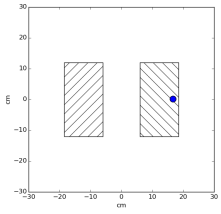

Figure 6.8: *ZMP when leaning to the right*

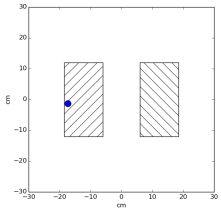

Figure 6.9: *ZMP when leaning to the left*

6.3 PID control

Once the ZMP has been successfully calculated, the next step is to see how the PID control works. In order to check the correct performance of the controller, the initial parameters have been used, although they are going to be modified later during the final tests. The controller gives an output depending on the computed ZMP while leaning the robot back and forth. This output is the velocity in degrees per second to be sent to the joint's motor.

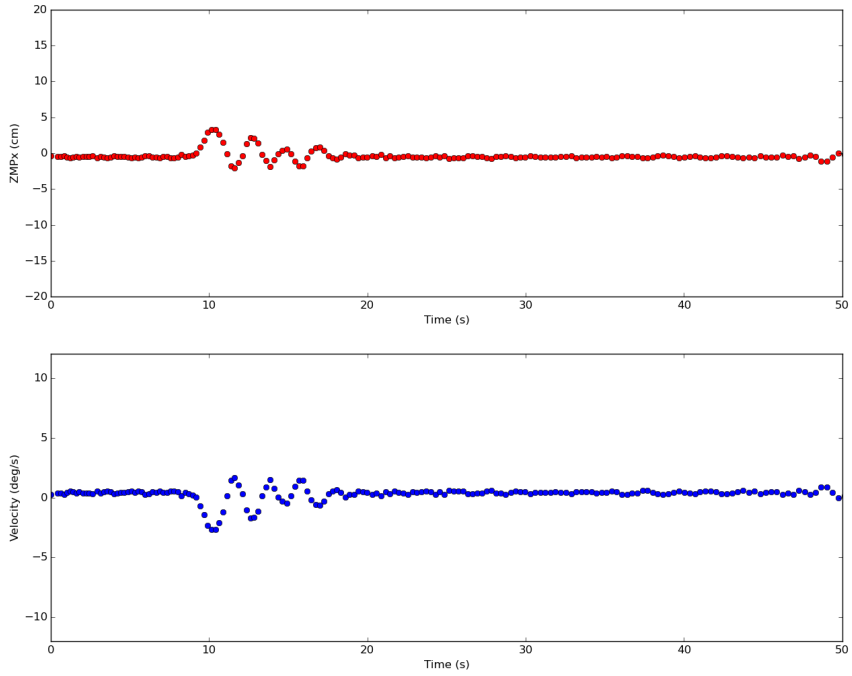

Figure 6.10: *PID output graph*

The calculated ZMP and the correspondent PID output have been graphically compared as shown in figure 6.10.

6.4 Joints motion

Before performing the final experiments, the independent movement of the was tested. This was performed on the robot's elbows just for convenience, and if the test was successful, then moving on to the ankles. The first try involved only one elbow. I was able to send a velocity value to a specific motor and it acted consequently. For this, I connected to a concrete YARP port (*/teo/rightArm*) and sent the appropriate command (*velocityMove*) indicating the two needed parameters: motor number and the velocity in degrees per second. After checking that the joint had moved correctly, the next attempt was to move both elbows at the same time. The procedure is the same but connecting to both YARP ports (*/teo/rightArm* and */teo/leftArm*) and sending two equal velocity commands. The result was successful.

```
Starting /home/teo/git/TEO_push/build/TEO_push...
yarp: Port /juan/leftArm/rpc:o active at tcp://2.2.2.50:10019
yarp: Port /juan/leftArm/command:o active at tcp://2.2.2.50:10020
yarp: Port /juan/leftArm/state:i active at tcp://2.2.2.50:10021
yarp: Port /juan/leftArm/stateExt:i active at tcp://2.2.2.50:10022
yarp: Sending output from /juan/leftArm/rpc:o to /teo/leftArm/rpc:i using tcp
yarp: Sending output from /juan/leftArm/command:o to /teo/leftArm/command:i using udp
yarp: Receiving input from /teo/leftArm/state:o to /juan/leftArm/state:i using udp
yarp: Receiving input from /teo/leftArm/stateExt:o to /juan/leftArm/stateExt:i using udp
[ok]
[INFO]created device <remote_controlboard>. See C++ class yarp::dev::RemoteControlBoard for documentation.
yarp: Port /juan/rightArm/rpc:o active at tcp://2.2.2.50:10015
yarp: Port /juan/rightArm/command:o active at tcp://2.2.2.50:10016
yarp: Port /juan/rightArm/state:i active at tcp://2.2.2.50:10017
yarp: Port /juan/rightArm/stateExt:i active at tcp://2.2.2.50:10018
yarp: Sending output from /juan/rightArm/rpc:o to /teo/rightArm/rpc:i using tcp
yarp: Sending output from /juan/rightArm/command:o to /teo/rightArm/command:i using udp
yarp: Receiving input from /teo/rightArm/state:o to /juan/rightArm/state:i using udp
yarp: Receiving input from /teo/rightArm/stateExt:o to /juan/rightArm/stateExt:i using udp
[success] TEO_push acquired robot left arm IVelocityControl interface
[INFO]created device <remote_controlboard>. See C++ class yarp::dev::RemoteControlBoard for documentation.
yarp: Removing output from /juan/rightArm/rpc:o to /teo/rightArm/rpc:i
yarp: Removing output from /juan/rightArm/command:o to /teo/rightArm/command:i
yarp: Removing input from /teo/rightArm/state:o to /juan/rightArm/state:i
yarp: Removing input from /teo/rightArm/stateExt:o to /juan/rightArm/stateExt:i
yarp: Removing output from /juan/leftArm/rpc:o to /teo/leftArm/rpc:i
yarp: Removing output from /juan/leftArm/command:o to /teo/leftArm/command:i
yarp: Removing input from /teo/leftArm/state:o to /juan/leftArm/state:i
yarp: Removing input from /teo/leftArm/stateExt:o to /juan/leftArm/stateExt:i
[success] TEO_push acquired robot right arm IVelocityControl interface
/home/teo/git/TEO_push/build/TEO_push exited with code 0
```

Figure 6.11: *Elbow joints test*

6.4.1 Problems found

YARP version

When trying to connect to the robot's devices, an error appeared indicating a problem with the *remotecontrolboard* protocol version. This happened because the PC and the robot's CPU worked on different YARP versions. The solution was easy: upgrade both to the last YARP version released.

Loop interval time

Once the previous problem was solved, a new one was found. While the program was running, the joint didn't move, but when it stopped, the velocity command affected the joint. After performing different tests changing some values, it was found that the problem was the loop interval time. It was set on 10 ms and it seemed to be too low, which means that the loop iterations were too fast. After changing it to 50 ms the program started working correctly.

6.5 Push recovery experiments

The push recovery experiments consisted in applying a disturbance (push) the robot and measuring the response, as well as visually checking the motion of the robot. The robot was pushed from both sides and in both planes. Sometimes the disturbance was a punctual push and other times it was a continuous push during two or three seconds (step actuation), where you could feel the resistance of the robot.

6.5.1 Sagittal Ankle Strategy

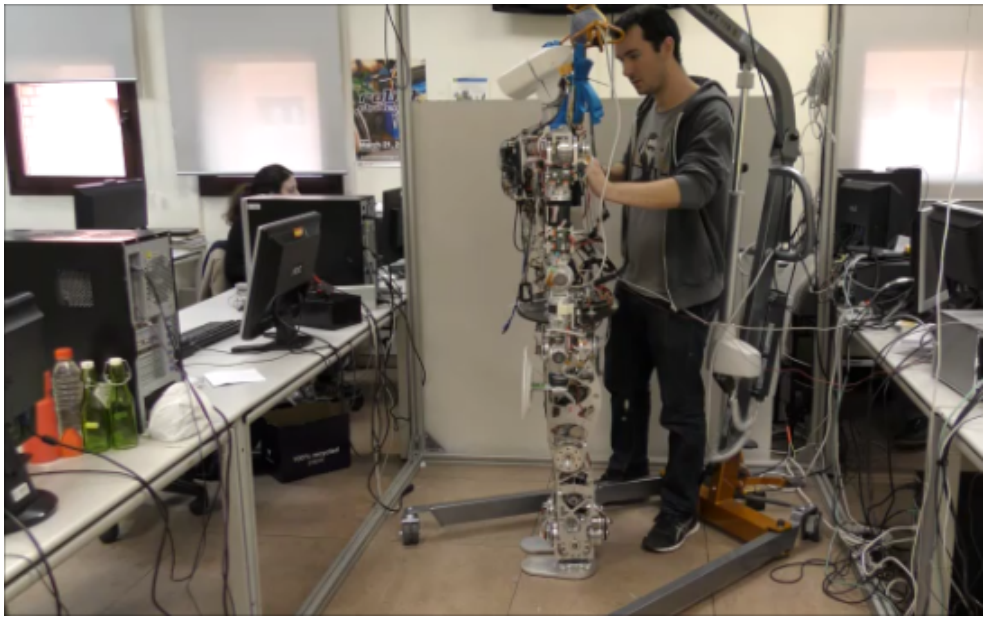

Figure 6.12: *Sagittal Ankle Strategy test*

Push-recovery tests were helpful for correctly designing the PID control. As the first design used didn't include derivative action, both proportional and integral terms increased overshoot significantly. But the integral action was needed in order to eliminate the steady-state error and also decrease the rise time, so the derivative action was added for reducing overshoot. The adjustment of the PID parameters has been done by experimenting and taking into account the overshoot, the rise time, the settling time and the steady-state error.

Moreover, another value to be tested was the number of samples used for the LP filter. As previously mentioned, using 10 samples seemed enough for a first attempt. Tests were made performing push recovery by Ankle Strategy and using different number of samples for the filter.

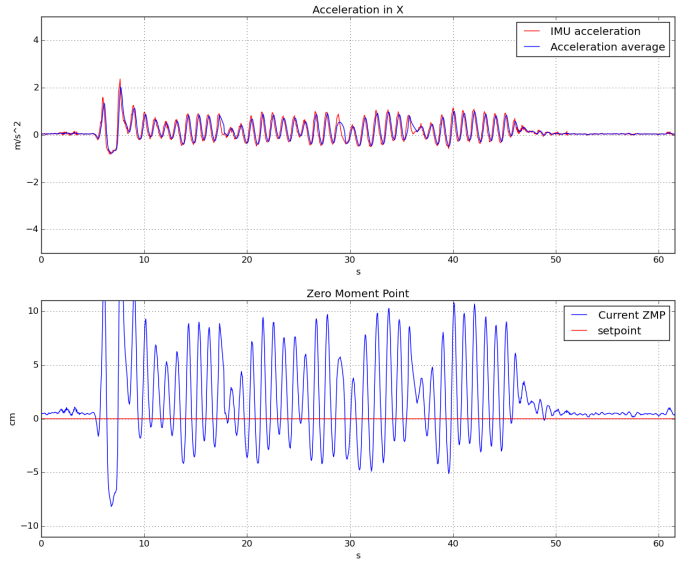

Figure 6.13: *Sagittal Ankle Strategy test results with LP filter using 5 samples*

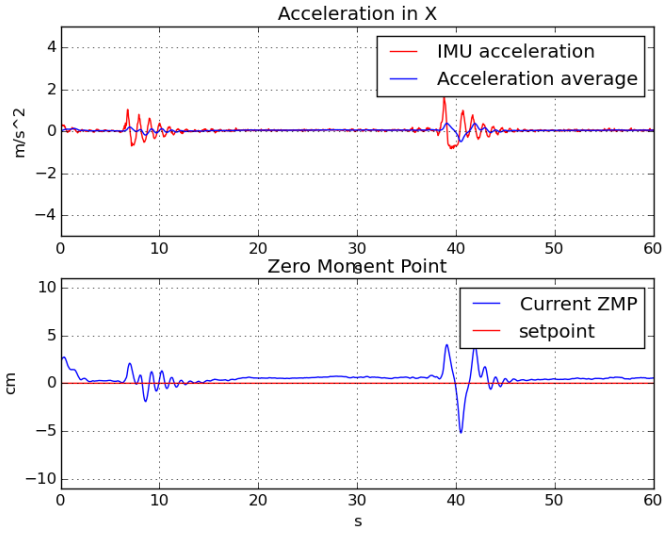

Figure 6.14: *Sagittal Ankle Strategy test results with LP filter using 30 samples*

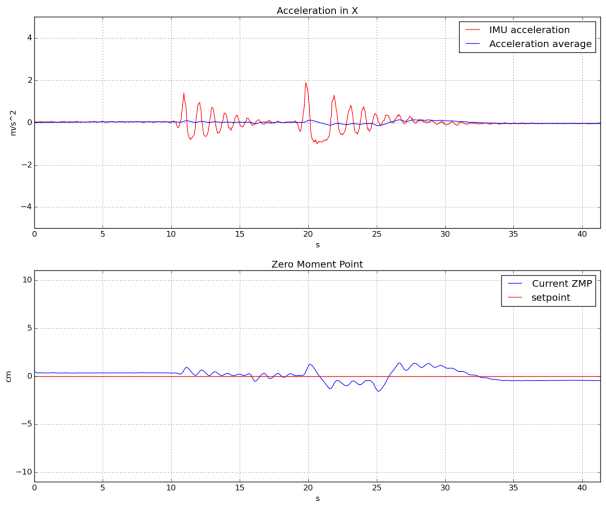

Figure 6.15: *Sagittal Ankle Strategy test results with LP filter using 100 samples*

Using just 5 samples (figure 6.13) was clearly not enough. It can be seen the high overshoot when trying to reach the setpoint after a disturbance. With 30 samples (figure 6.14) it seemed to work fine. Enough for slightly smoothing the accelerations, and the system reached the setpoint in an acceptable time, with a negligible error (< 1 cm). When using 100 samples for the filter (figure 6.15), the average smoothed too much the accelerations. The system's settling time is too high. It was concluded that the following values were acceptable for a valid Ankle Strategy:

- Kp = 0.1
- Ki = 0.001
- Kd = 0.01
- Filter samples = 30

Problems found

A significant problem found was related to the use of an external Python application for plotting the results of the test. Sending data through a YARP port to this external application caused a considerable delay in the main program. The problem was solved changing the way of plotting the results: instead of using YARP ports, the main program save the relevant data in an external text file for plotting after stopping the program. However, saving data in an external file requires computation time, and this is why this process is only done for checking results, but won't be implemented in TEO's final software.

6.5.2 Strategy determining

This test consisted in pushing the robot and checking if the strategy is correctly chosen.

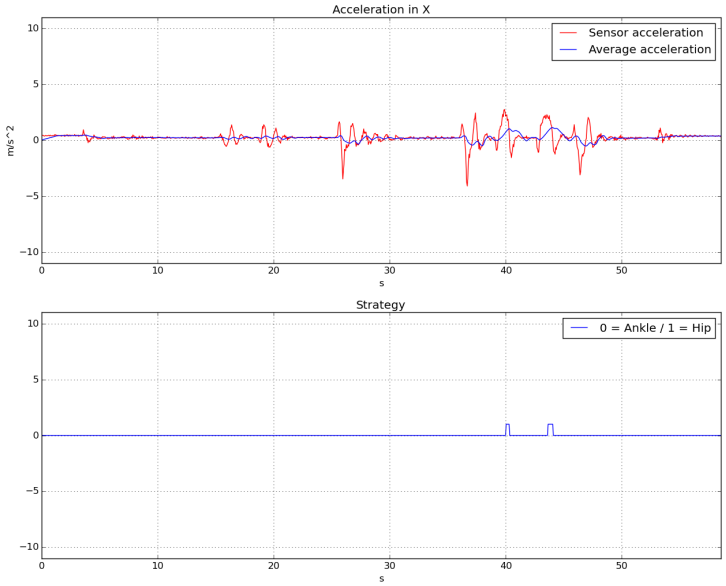

Figure 6.16: *Strategy determining test results with 30 samples*

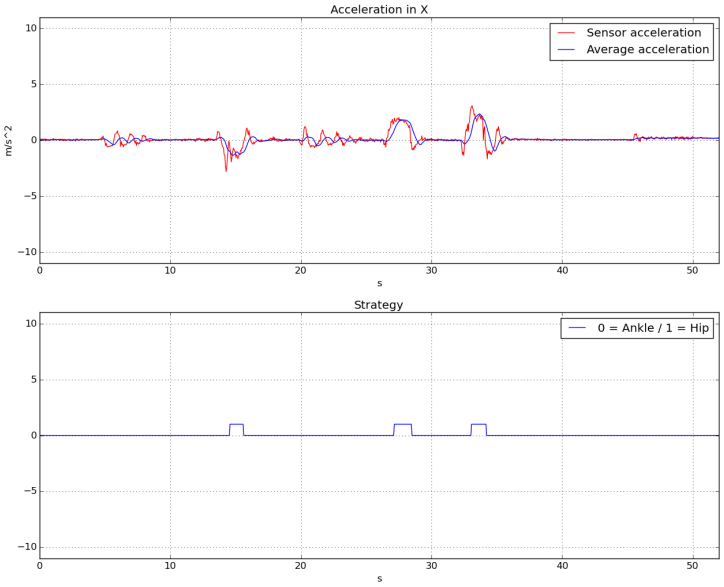

Figure 6.17: *Strategy determining test results with 15 samples*

Using 30 samples for the LP filter (figure 6.16) softened the acceleration average too much. When the robot experienced a disturbance too high for the ankle, the strategy was incorrectly chosen. To solve this, the new value chosen was half of it: 15. As it can be seen in figure 6.17, when the disturbance is too high for the Ankle Strategy, the Hip Strategy is chosen.

6.5.3 Sagittal Hip Strategy

The Hip Strategy test involved the previous strategy determining and also performing the chosen strategy correctly. As explained in the previous chapter, in this strategy the hip joins the ankles in an attempt to counteract the fall. Another method was tested in which the hip counteracted the fall but the ankles counteracted the effect of the hip in order to handle the inertial forces created by its movement. However, the original approach proved to be the most effective.

A problem appeared when trying to do the test: the velocity control of the hip's joint wasn't working. This problem is explained later in the "problems found" subsection. In order to try the Sagittal Hip Strategy, a position control was implemented instead. The first attempts showed that the hip's reaction movement wasn't big enough and the robot couldn't recover its position. This was solved increasing the proportional parameter from to 0.1 to 1.

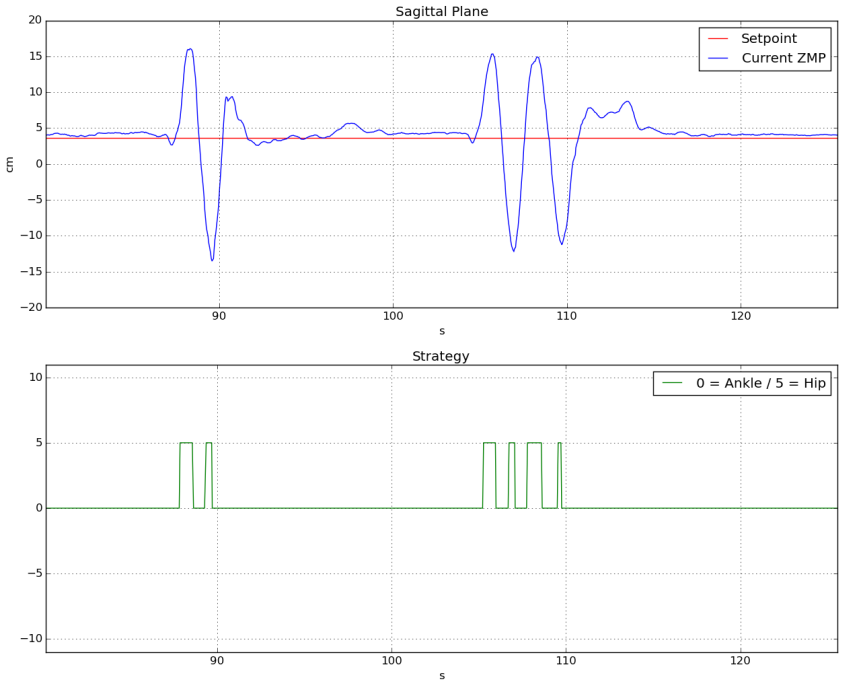

Figure 6.18: *Sagittal Hip Strategy test results*

As the results in figure 6.18 show, when the disturbance is too high for the Ankle Strategy, the Hip Strategy is chosen and the hip reacts in order to keep balance. Due to the problem with the velocity control, the Hip Strategy only works fine with not very strong disturbances. However, once the problem is solved, it should work correctly commanding velocity instead of position.

Problems found

Velocity control not working on TEO's hip

The worst problem appeared when we found out that the velocity control for joint number 14 (figure 4.5) wasn't working. The joint could be moved using the position control but not by commanding a velocity. For the correct performance of this equilibrium control, we need to be able to command velocities, something that is possible for the rest of the joints and has been done throughout all the project. The problem with commanding a position instead of a velocity is that the movement of the joint is always at a constant velocity. This means that when the stable position is being recovered the hip doesn't slow down as it would do using the PID output in a velocity control. The consequence is that the COM's linear velocity is always high and this causes that, according to equation (3.5), the Hip Strategy is always needed. In other words, the step regarding the determining of the strategy needed uses the COM's velocity to do so, and if it is always high then the system determines that the hip is always needed. The experimental consequence of all this is that since the first time that the hip moves, it keeps moving back and forth relentless.

TEO's physical components (driver, encoder...) were revised, as well as TEO's software, but we still don't know what is the problem and can't fix it. Other options were tried using the position controller, but none worked fine.

6.5.4 Frontal Strategy

A test was made to check the correct performance of the balance strategy for the frontal plane, including the control of the four joints involved. During these experiments, we were able to contrast the results and use this information to choose valid parameters for the PID controller (Kp, Kd and Ki).

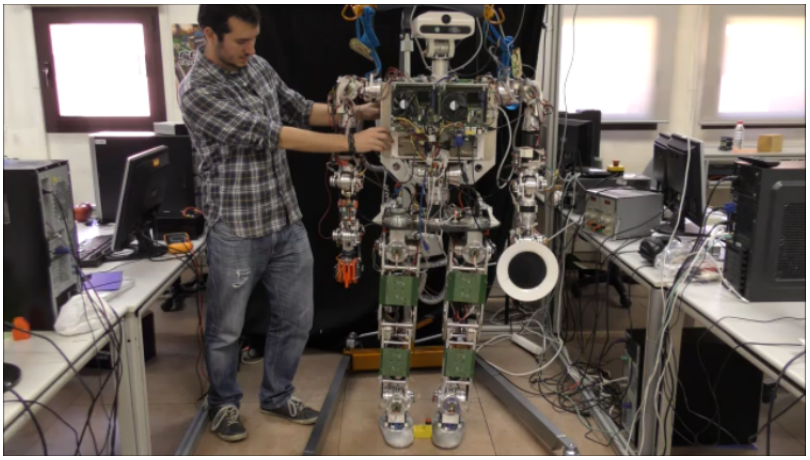

Figure 6.19: *Frontal Strategy test*

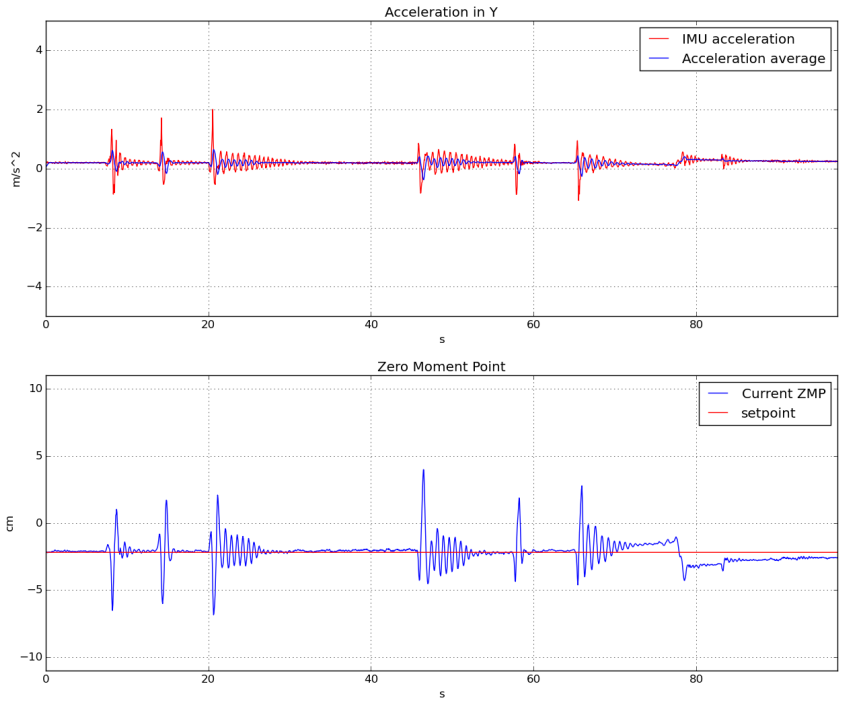

Figure 6.20: *Frontal Strategy test results*

Performing this test with the same parameters as in the Sagittal Ankle Strategy (Kp = 0.1, Kd = 0.01, Ki = 0.001) had successful results. As it can be seen in figure 6.20, the robot was able to recover its position by itself, avoiding the fall. Sometimes it does so with more overshoot than other times. This depends on the design of the PID. The irregularity at the end of the graph corresponds to the moment when the robot is lifted by its crane before finishing the experiment, for safety reasons.

6.5.5 2D Equilibrium Control

The final equilibrium control test includes everything seen previously. It involves the determining strategy step and the performance of the three implemented strategies: Sagittal Ankle Strategy, Sagittal Hip Strategy and Frontal Strategy.

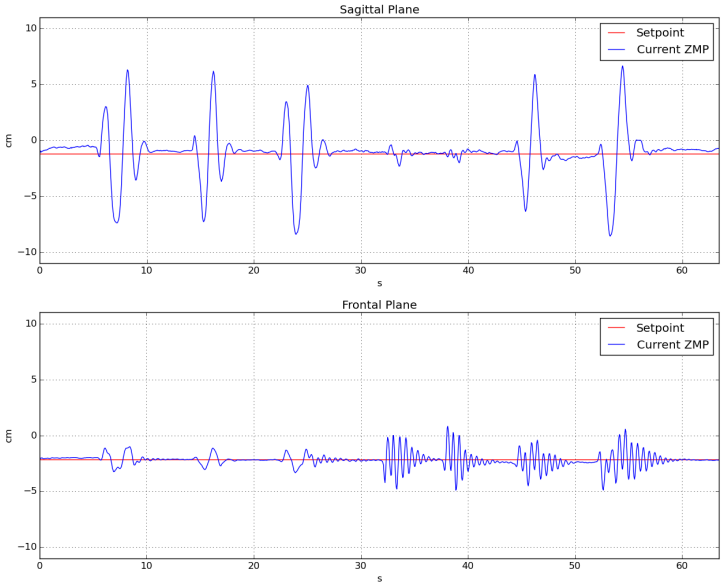

Figure 6.21: *2D Equilibrium Control test results without Hip Strategy*

Firstly, the 2D Equilibrium was tested without implementing the Hip Strategy, only using the ankles for both sagittal and frontal planes. During the first 30 seconds, three pushes were applied only in the sagittal plane. For the next 10 seconds, two pushes were applied but now in the frontal plane. Finally, for the last 20 seconds, two diagonal pushes were applied, this time involving both planes. As figure 6.21 shows, the robot was able to stabilize itself and return to the setpoint. Then, the next step consisted in adding the Hip Strategy to the 2D Equilibrium Control. Results (figure 6.22) show that the strategy is correctly chosen and the robot is able to stabilize. However, as explained previously, using the position controller for the hip makes the Hip Strategy only work fine when the disturbances are not too high. Besides that, during the final push recovery experiments, we were able to experimentally prove that the loop interval time is in fact 50 ms, as expected, as well as obtain other relevant data (iteration time, error in the ZMP...)

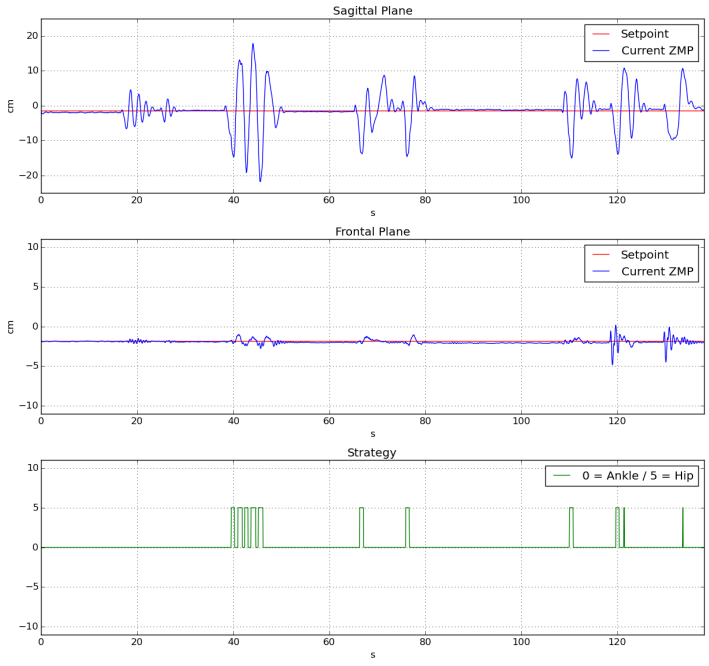

Figure 6.22: *2D Equilibrium Control test results*

Problems found

Transmission of commands too fast

While the control was running, the frontal thread suddenly stopped. It happened again and again, and at different times. It was found that the problem was that there was not enough delay between the velocity commands. The transmission of commands was too fast and this caused problems in the communication with the robot devices. The *velocityMove* function includes an internal delay, but it seemed not to be enough when several commands are sent one after the other during a short period of time. It started working fine after a low extra delay was added between the velocity commands.

Chapter 7

Conclusions

After analyzing all the experiments, specially the final push recovery with the 2D equilibrium control, I can assert that the results are satisfactory. The equilibrium control works almost perfectly in both planes, although the PID controller still can be improved.

When performing the Frontal Strategy and the Sagittal Ankle Strategy, the robot is able to stabilize with little overshoot. The result is very good. It is a fine control for small disturbances.

The only arguable point is the Sagittal Hip Strategy due to the issue with the velocity control for the hip joint. Because of that problem, this strategy only works fine when the disturbance is not too high. However, once this problem is solved and then controlling velocity, it should work correctly for larger disturbances, or at least it should be easy to improve and achieve a satisfactory control.

The final system's parameters and characteristics are shown in table 7.1.

Ankles PID	**Kp**	0.1
	Kd	0.01
	Ki	0.001
Hip PID	**Kp**	1
	Kd	0.01
	Ki	0.001
Low Pass Filter Samples		20
Iteration Time		$\sim$ 20 ms
Loop Interval Time		50 ms
ZMP Error	**x**	$\pm$ 0.22 cm
	y	$\pm$ 0.1 cm

Table 7.1: *Summary of the system's characteristics*

Chapter 8

Future projects

8.1 Improve PID design

Tuning the PID controllers correctly would make the robot stabilize faster, with less error, with less overshoot, etc. This is a matter of finding the desired values for the controller's parameters (Kp, Ki and Kd) as well as for the number of samples used in the low pass filter.

8.2 Implementation of other strategies

8.2.1 Counterbalance using arms

If there is a need to develop more effective controls, a factor worth to take into account is the use of the arms to counterbalance. Moving the arms back or forth helps keeping balance by changing the center of gravity. However, this can't be implemented if the robot is supposed to manipulate.

8.2.2 Knee bending

Bending the knees to keep balance is inherent in humans, as it helps to recover stability. In this case, we would be using also joints 3 and 10 (figure 4.5). Authors have already studied this process in biped robots (Ono, Sato, & Ohnishi, 2011).

8.2.3 Use of sagittal hip joints

As mentioned at the introduction to this thesis, in this project we are referring as the hip joint to what actually is the waist joint. The hip sagittal joints (4 and 9 in figure 4.5) can be also used in the sagittal strategies in order to improve the control.

8.2.4 Stepping

The implementation of the third balance strategy (Stephens & Atkeson, 2010b), which consists in avoiding the fall by making a step, has not being possible due to the stepping motion of robot TEO, which is still in development. In fact, at this moment, TEO can't perform a step as fast as it would be needed to avoid the fall. It is an interesting and complex project to carry out in the future.

8.3 Manipulation stabilization

Once the robot is able to balance when standing still, an interesting new project would be to use this advances for keeping balance during the manipulation of objects. The procedure may be similar but you have to take into account possible desired changes in accelerations, velocities or the ZMP.

8.3.1 Manipulation of Heavy Objects

The process of picking up a heavy object involves extra forces disturbing the balance (Stephens & Atkeson, 2010a). Those forces need to be taken into account for the equilibrium control for avoiding the fall. Not only this is important when picking it up, but also during the manipulation of those heavy objects.

8.3.2 Fast-motion manipulation

Even if the object being manipulated is not heavy, certain manipulation tasks have a high influence in equilibrium. The most clear examples are those tasks in which the robot moves one of its articulations very fast. When doing that (e.g. for a long ball toss), the inertia produced creates a disturbance in the whole system, which needs to be counteracted with the equilibrium control.

8.3.3 Waiter Robot

A project being carried out by the RoboticsLab consists in a Waiter Robot (Lorente, García, Martínez, Hernández, & Balaguer, 2016). It is an assistive humanoid robot developed in TEO's platform able to perform tasks corresponding to waiters. In this case, it is more complex because not only there must be an equilibrium control for the whole-body balance, but also for the objects being carried in the tray that the robot is provided with.

8.4 Bipedal locomotion stabilization

The aim of this project would be to preform an equilibrium control during locomotion, so the robot can walk and keep balance at the same time. I think that the key here is to modify the setpoint for the PID controller in relation to the humanoid locomotion. If the gait is planned according to the calculated ZMP evolution (Arbulu & Balaguer, 2007), then this constantly changing ZMP would be the setpoint for keeping balance.

References

Čapek, K. (1923). *R.U.R. (Rossum's Universal Robots)* (N. Playfair, P. Selver, & W. Landes, Eds.). Garden City, NY, USA: Doubleday, Page & Company.

Arbulu, M., & Balaguer, C. (2007). Real-time gait planning for Rh-1 humanoid robot, using Local Axis Gait algorithm. In *7th IEEE-RAS International Conference on Humanoid Robots* (pp. 563–568). Pittsburgh, PA, USA.

Asimov, I. (1950). *I, Robot*. USA: Gnome Press.

Assman, T. M. (2012). *Humanoid push recovery stepping in experiments and simulations* (MSc Thesis). Technische Universiteit Eindhoven, Eindhoven, Holland.

cplusplus. (2016). Retrieved 2016-04-11, from `http://www.cplusplus.com/`

Hofmann, A. G. (2006). *Robust execution of bipedal walking tasks from biomechanical principles* (PhD Thesis). Massachusetts Institute of Technology, Boston, MA, USA.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95.

Kajita, S., & Espiau, B. (2008). Legged Robots. In B. Siciliano & O. Khatib (Eds.), *Springer Handbook of Robotics* (pp. 361–389). Berlin, Germany: Springer-Verlag.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)* (Vol. 2, pp. 1620–1626 vol.2). Taipei, Taiwan.

Kajita, S., & Tani, K. (1991). Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In *Proceedings of the 1991 IEEE Intemational Conference on Robotics and Automation (ICRA 1991)* (pp. 1405–1411 vol.2). Sacramento, CA, USA.

Lang, F. (1927). *Metropolis.* U.F.A.

Lorente, J., García, J. M., Martínez, S., Hernández, J., & Balaguer, C. (2016). Waiter Robot: Advances in Humanoid Robot Research at UC3m. In *RoboCity16: Open Conference on Future Trends in Robotics* (pp. 195–202). Madrid, Spain.

Martínez, S. (2012). *Human inspired humanoid robots control architecture* (PhD Thesis). Universidad Carlos III de Madrid, Madrid, Spain.

Metta, G., Sandini, G., Vernon, D., Natale, L., & Nori, F. (2008). The iCub Humanoid Robot: An Open Platform for Research in Embodied Cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems (PerMIS '08)* (pp. 50–56). Gaithersburg, MD, USA.

Monje, C. A., Martínez, S., Jardón, A., Pierro, P., Balaguer, C., & Muñoz, D. (2011). Full-size humanoid robot TEO: Design attending mechanical robustness and energy consumption. In *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (pp. 325–330).

Ono, H., Sato, T., & Ohnishi, K. (2011). Balance recovery of ankle strategy: Using knee joint for biped robot. In *1st International Symposium on Access Spaces (ISAS)* (pp. 236–241). Yokohama, Japan.

Petrovic, V., Jovanovic, K., & Potkonjak, V. (2014). Influence of external disturbances to dynamic balance of the semi-anthropomimetic robot. *Serbian Journal of Electrical Engineering*, *11*(1), 145–158.

Python. (2016). Retrieved 2016-04-11, from `https://www.python.org/`

Ruina, A., & Pratap, R. (2015). *Introduction to Statics and Dynamics*. Oxford, England, UK: Oxford University Press.

Sardain, P., & Bessonnet, G. (2004). Forces acting on a biped robot. Center of pressure - Zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *34*(5), 630–637.

Snyder, B. (2015). *PID C++ implementation.* Retrieved 2015-11-17, from `gist.github.com/bradley219/5373998`

Stephens, B. J. (2011). *Push Recovery Control for Force-Controlled Humanoid Robots* (PhD Thesis). Carnegie Mellon University, Pittsburgh, PA, USA.

Stephens, B. J., & Atkeson, C. G. (2010a). Dynamic Balance Force Control for compliant humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1248–1255). Taipei, Taiwan.

Stephens, B. J., & Atkeson, C. G. (2010b). Push Recovery by stepping for humanoid robots with force controlled joints. In *Proceedings of the 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2010)* (pp. 52–59). Nashville, TN, USA.

Vadakkepat, P., & Goswami, D. (2008). Biped Locomotion: Stability, Analysis and Control. *International Journal on Smart Sensing and Intelligent Systems*, *1*, 187–207.

XsensTechnologies. (2009). *MTi and MTx User Manual and Technical Documentation.*

XsensTechnologies. (2011). *MTi, Miniature Attitude and Heading Reference System.*

YARP: Yet Another Robot Platform. (2016). Retrieved 2016-04-11, from `http://www.yarp.it/`

Appendix A

Guide and Code

Follow these steps for launching the equilibrium control module:

- **Terminal 1:**
  ssh 2.2.2.52 / ssh locomotion
  yarpdev - -subdevice xsensmtx - -device inertial - -name /inertial - -verbose
- **Terminal 2:**
  ssh 2.2.2.52 / ssh locomotion
  launchLocomotion
- **Launch TEO_push**

Full code:
`https://github.com/j-lorente/TEO_push.git`

Acknowledgements to:
`https://github.com/roboticslab-uc3m/teo-body.git`

PID C++ library from (Snyder, 2015).