

Humanoid Locomotion Planning
for Visually-Guided Tasks

Thesis

TO ACHIEVE THE ACADEMIC DEGREE OF

Master of Science in Mechatronics

BY

Tonatiuh Saldaña Mora

Saltillo, Coahuila, México, September 2011

To Andrea and Aide

Acknowledgements

I dedicate this Master thesis work, which has been developed due to the will and benediction of God, to my daughter Andrea Victoria, whose sweet smile gives me each day the inspiration in the mission of constructing a better world for our children; and to my wife Aide Yeline, whom during these last years and with absolute love, generously withstood all the deposited efforts to accomplish the goals of the present project, from its very beginning.

I would like to give special thanks to my parents, Othón and Victoria, whose love and wise teachings have been crucial for my formation; also to my brother Huitzilhuitl, whom has been like a mentor since I was a little child; and to my grandmothers Amelia and Celia, whom with their almost a hundred years age, still keep blessing my spirit and encouraging my daily quests.

It is an honor to thank Dr. Gustavo Arechavaleta, who has been my tutor for the development of the present project and has also placed on me the motivation to commence a career in the world of scientific research.

I would also like to thank Dr. Ernesto Olguín, Dr. Jean-Bernard Hayet, Dr. Claudia Esteves, M.C. Jerardo Jarquín, M.C. Pablo, M.C. Arturo Barrios and Ing. Júpiter Barrera, whose backup has been transcendental in the attempt to build on a project with a world-class quality.

Big thanks to the staff of CIDESI Queretaro, FH-Aachen, CINVESTAV Saltillo, CIMAT Guanajuato and also to CONACYT for their support and trust.

To my dear comrades Héctor, Ricardo, Carlos, Ariadna, Rodolfo, José, César, Jorge, Daniel, Enrique, Damián, Hugo, Zuleima, Alejandro, Diana, Emanuel, Benítez and Rubén, I would like to give enormous thanks for their unconditional friendship and encouragement during my whole life.

Abstract

This master thesis work presents the theoretical fundamentals, development strategies and implementation results of a walking pattern generator for humanoid robots, which is suitable for visually-guided tasks.

In first place, a general introduction about the state-of-the-art related to research on humanoid locomotion solutions is introduced, as well as a general overview about humanoid robots, making emphasis on the main characteristics that identify this type of robots from the rest of the existing platforms.

Furthermore, a method to construct the kinematic model for robots based on a tree-like structure is proposed. Subsequently, several algorithms and relations necessary to compute the model forward kinematics and to calculate the different versions of Jacobian matrices are introduced.

Moreover, this work presents and explains two powerful techniques to compute the model inverse kinematics. These methods take advantage of the inherent redundancy of humanoid robots to simultaneously solve a determined amount of kinematic tasks, based on a prioritized scheme.

Later, all important fundamentals related to the humanoid dynamics are introduced. Particularly, the concept of the Zero-Moment Point (ZMP), which is a key parameter on humanoid locomotion, is presented in this thesis work and besides, some methods to determine the position of this parameter are discussed as well.

This thesis also presents some schemes to model the process of biped locomotion, focusing on a method to determine the optimal trajectory of the robot Center of Mass (CoM), based on the preview control of the reference ZMP position.

Besides, all fields that are studied in this thesis are taken together into a global architecture for walking pattern generation, consisting on two stages of control, each of them containing the processes of linear-quadratic regulation of the reference ZMP, Prioritized Inverse Kinematics (PIK) and approximation of the real position of the ZMP based on the robot movements.

A genuine verification of the functionality of the designed walking pattern generator is exposed at the end of this thesis work, by presenting the results of the performed simulations on a virtual humanoid model, as well as the experimentations on a physical humanoid robot.

Table of Contents

Chapter 1 Introduction	1
1.1 Background	1
1.2 Problem definition.....	3
1.3 Justification	4
1.4 Objectives.....	6
Chapter 2 Fundamentals	7
2.1 Concept of humanoid robot.....	8
2.2 History of humanoid robots	10
2.3 Constitution of humanoid robots.....	12
2.3.1 Robot geometry, bodies and articulations	12
2.3.2 Mechanical structure of a humanoid robot and its representation.....	14
2.3.3 Actuators, movement transmissions and reducers used in humanoid robots.....	16
2.3.4 Sensors used in humanoid robots	17
2.3.5 Control system of a humanoid robot	20
2.4 Important aspects in humanoid robots	22
2.4.1 Degrees of freedom in humanoid robots and concept of redundancy.....	22
2.4.2 Humanoids as mobile robots and concept of non-holonomy	23
2.4.3 Underactuated nature of humanoid robots	24
2.5 Kinematics of Humanoid Robots	25
2.5.1 Coordinate system frames	25
2.5.2 Generation of the kinematic model	26
2.5.3 Representation of the position and orientation in the space of a rigid body	29
2.5.4 Forward Kinematics	32
2.5.5 Differential kinematic modeling and the Jacobian matrix	37
2.5.6 Inverse Kinematics	47
2.6 Dynamics of humanoid robots	61
2.6.1 Basic Concepts in robot Dynamics.....	61
2.6.2 The Zero-Moment Point (ZMP).....	66
2.6.3 Determination of the position of the ZMP	69
2.7 Walking Pattern Generation	75
2.7.1 Generation of a walking pattern based on the 3D-LIP.....	75
2.7.2 Generation of a walking pattern based on the ZMP.....	78
2.7.3 Walking Pattern Generation based on the Preview Control of the ZMP	81
Chapter 3 Methodology	86
3.1 Development strategies	87
3.2 Mathematical representation of kinematic tasks	89
3.2.1 Position and Orientation	89
3.2.2 Articular Limits	90
3.2.3 Obstacle Avoidance.....	90
3.2.4 Static Balance	91
3.3 Global Architecture of the Walking Pattern Generator	94
3.4 Determination of the feet trajectories and head orientation.....	99
3.4.1 Feet trajectories.....	99
3.4.2 Head Orientation.....	103

Chapter 4 Simulations, experiments and results.....	105
4.1 Walking forward through a straight line path	108
4.2 Walking backwards through a straight line path.....	111
4.3 Walking forward through a curve line path	113
4.4 Rotating around the vertical axis.....	116
Chapter 5 Conclusions, Recommendations and Future Work	119
5.1 Conclusions	119
5.2 Recommendations and Future Work.....	120
Bibliography	121
Appendix A: Convention of Euler Angles	124
Appendix B: Equation of Rodrigues	126
Appendix C: Definition of Convex Hull	127
Appendix D: Solution of the discrete-time algebraic Riccati equation by using RSF representation	128
Appendix E: General description of the NAO platform.....	129

Chapter 1

Introduction

1.1 Background

Research in anthropomorphic robots is currently one of the most popular and interesting topics in the field of Robotics. Scientists and engineers are now developing important projects related to this kind of systems, such as biped locomotion, human-robot interaction, stimulus response, vision and audio recognition, material handling and task performance. A major example of the intensive quest to improve the design of anthropomorphic interfaces is the development of humanoid robotic platforms. During the last years, a great amount of humanoid robots have been constructed and, for each of these, much scientific literature is available describing their motion engine and the algorithms to create different walking patterns.

Considering the task of autonomous motion planning, although many of the developed works have already demonstrated very reliable dynamic biped walking, it is still important to understand the theoretical background of biped locomotion and therefore several improvement features should be developed as well.

Locomotion in human world is an essential motion skill for human-robot interaction tasks. Today's requirements imply that robots are able to move and interact with the environment. Thanks to substantial contributions in humanoid motion planning, it is currently possible to make use of powerful motion planners, in order to solve humanoid whole-body motion problems. However, biped locomotion is a more complex case of motion planning, due to the inherent redundancy of human-like kinematic structures, the unavoidable dynamic constraints and their subactuated nature.

Most of the proposed humanoid locomotion planners reported in the available literature are based on a two-stage strategy. At the first stage, a global path linking the initial and final configuration (i.e. position and orientation) is constructed using a simplified model of the system. Once a global path is found, then, at the second stage the motion coordination of the whole body is generated. A key component of the planner is the local method which allows joining a pair of configurations in the free configuration space.

Additionally, a series of control strategies based on generalized inverse pendulum models have been proposed to control the motion of the center of mass (CoM) within a horizontal

plane in order to make the Zero Moment Point (ZMP) follow a prescribed trajectory defined by successive planned steps. Thus, the available pattern generators are based on a control solution, by which from the footsteps given as an input, the CoM trajectory is generated using a 3D linear inverted pendulum model of the robot whose CoM moves on a plane. The key of this strategy is to solve an inverse problem from the ZMP reference position deduced from the footsteps. This inverse problem is solved using a preview controller [1] and therefore implies that the system has knowledge about the near future in the corresponding window.

In order to take into account the real model of the robot, the field expert Dr. Eng. Shuuji Kajita et al., proposed to use a second stage of preview control to compensate the difference between the ZMP of the multibody model and the ZMP of the inverted pendulum. Nevertheless, even though this second stage of preview control is extremely efficient, it is not able to provide the enough amount future knowledge needed for the system locomotion requirements.

On the other hand, in order to satisfy basic requirements in humanoid robot capabilities, humanoid robots are yet to be provided with a wide range of capabilities, such as autonomous learning attributes, self maintenance, and safe interaction with environment. Biped locomotion should be performed with a visual guidance, so the robot is able to identify paths, targets or obstacles while walking.

So far, not much research has been developed for sensor-based control of whole-body humanoid robots. Some works have been proposed to solve visual guided tasks in anthropomorphic platforms; for instance, K. Yamamura and N. Maru developed a visual servoing technique for leg positioning and other researchers implemented vision-based manipulation systems for a non-walking humanoid torso. These works demonstrate the efficiency of sensor-based reactive control for developing robust and accurate task for humanoid robots. However, none of them was extended for full-body motion generation.

Being able of performing different tasks while walking is currently an important area of research in humanoid Robotics. An effective combination of biped locomotion and visual guidance will provide humanoid robots a wide range of capabilities and furthermore, will open a transcendental window to achieve more ambitious and useful applications, such as material handling, environmental interaction, decision making and many others.

1.2 Problem definition

During the last decade, important progress in the field of Robotics has been reached due to the increasing advances in the study, construction and improvement of anthropomorphic structures and, furthermore, humanoid robots. Nevertheless, much work is yet to be done in order to optimize these systems performance.

Concerning this master project, the whole problem to solve is to find out **how is it possible to autonomously transport a biped humanoid robot from a first to a final configuration (x, y, θ) on a previously known planar surface, while simultaneously maintaining dynamic balance, avoiding obstacles and visually controlling the body orientation heading to a fixed landmark.**

After reading the previous problem statement, one can realize that there are basically three main challenges are to be attended and, as a combination of these, an investigation problem is to be solved. These challenges are, in the first place, **walking pattern generation**, where a robust motion planner is to be implemented, taking into account the developed studies available in the state of the art; in second place, **Kinematics**, where an accurate model is to be constructed and a series of solutions to compute the robot forward and inverse kinematics are to be implemented; and finally, **Dynamics**, providing the knowledge of parameters of relevance in humanoid locomotion. Even though each of these aspects will be treated individually in the first stage of the project, they should be considered together when attempting to give a final solution to the investigation problem.

1.3 Justification

The importance of researching about humanoid robotics resides, first of all, in increasing our knowledge about the human natural behavior, as it would be extremely difficult to implement, for instance, a motion generator on a robotic platform if one does not analyze the human body motion in the first place. Therefore, investigating and developing solutions on humanoid robots is an efficient way to know more about ourselves, and furthermore, about all the physic, chemic and mathematic laws which are involved in human physiology and govern our body's functions.

Future advances in the study of humanoid robotics would certainly bring important solutions for biomedical purposes as well, like the design and implementation of human prostheses for example. Prosthetics is actually a really ancient idea, there is also very efficient developed work and functional prototypes of these solutions are already available in medical facilities; nevertheless, there is still much work to be done concerning this topic and humanoid robotics play a very important role considering the series of components to integrate prosthetic limbs to the human body, like biosensors, data processors, controllers and actuators. Therefore this transcendental quest to give the handicapped a new opportunity to reintegrate themselves to their previous lives, takes humanoid robotics into an important rank, full of development possibilities.

On the other hand, the construction of autonomous humanoid platforms is gradually acquiring popularity for domestic, some industrial and personal assistance tasks. As a result of this, each time more humanoid robots are commercially available to perform cleaning, hosting, bar tending, nursing, cooking and also dirty or dangerous activities, like working in bio-hazardous environments, outer space, underwater or high mountain. Some repetitive jobs are also suitable for humanoid robots; therefore they are also being used in industrial manufacturing lines. These applications of humanoid robots represent a slightly benefic resource which is able to increase productivity, quality and safety, while simultaneously reducing costs and stimulating continuous improvement in world class enterprises.

Essentially, as humanoids can use tools and operate equipment and vehicles designed for the human anatomy, they could theoretically perform any task a human can, as long as it is provided with the proper hardware and software. Here is where the main challenge of research about human robotics resides. The more related to the real human behavior, the more complex the system becomes. Today, the main goal of the research in this field of interest is to provide a humanoid robot which is capable of walking in our quotidian environment, going up and downstairs, planning itineraries by itself, not being so damaged if it falls down and reincorporating by itself to a stand-up position, avoiding

obstacles, moving through an irregular surface, opening and closing doors, manipulating objects with its hands while maintaining equilibrium and communicating with people and other robots. All these particular problems require numerous studies before being completely solved. Thus, considering all aspects mentioned in this section, each progress obtained while investigating and developing works in humanoid platforms represents a step forward in the every day effort to bring new sources of health, happiness and richness for mankind.

1.4 Objectives

The general objective of this thesis project is **to implement a walking pattern generator for a humanoid robot for visually-guided navigation tasks**. By going deeper into this goal, it is possible to identify specific objectives which define particular actions that are going to be performed in order to achieve the stated general objective.

Thus, the specific objectives for this project are the following:

- Deduce a mathematic model which describes the physics of biped locomotion, considering a tree based kinematical structure moving on a plane from a first to a final configuration (x, y, θ)
- Develop a technique which converts the robot desired footsteps positions (which might be user defined or possibly obtained from a path planner) to a desired ZMP trajectory, based on given robot structural parameters.
- Implement a walking model based on the ZMP method and using Kajita's preview controller, which will transport the kinematical structure through the desired ZMP trajectory, maintaining dynamic balance and considering the natural equalities and inequalities, which are specific of humanoid robots.
- Integrate an inverse kinematics process in order to transform the obtained ZMP trajectory to angular displacements of the robot articulations.
- Include a subsystem to the process which controls the body and head orientations, allowing guidance through the defined path, so the robot can be able to maintain visual contact with a fixed landmark on the plane.
- Implement the developed solutions in a provided physical humanoid platform and perform corresponding tests.

This work deals with the complexity of whole-body humanoid motion and optimization algorithms. This will be done by considering humanoid properties, i.e. stable walking behavior and whole-body motion generation and control. As a consequence, this master thesis project will provide a unified framework for humanoid locomotion planning that incorporates visual landmarks. Thus, as mentioned, the general purpose of this thesis is to develop a walking pattern generator based on the Kajita's preview controller for landmark-based navigation tasks.

Chapter 2

Fundamentals

This chapter makes a grasping into the state-of-the-art and exposes the most important theoretical fundamentals about humanoid robots and particularly emphasizes on biped locomotion topics, based on official literature published by a series of experts on the field whom are currently working worldwide in the constant development of this area of study.

Section 2 presents, in a general way, the concept and the most relevant characteristics of humanoid robots, which make them different from other kinds of manipulators. Subsequently, in section 2.2, a brief review about the historical development of humanoid robots is introduced.

By its part, section 2.3 explains the constitution of humanoid robots, which in essence is not very different to other types of manipulator robots. This section also introduces the tree-like structural representation of humanoid robots, which is the basis for some of the algorithmic solutions exposed in this thesis work.

Moreover, section 2.4 introduces the concepts of redundancy, under-actuation and non-holonomy, which are important aspects which characterize the nature and behavior of humanoid robots. The consideration of these aspects is imperative for practical purposes.

All the theoretical fundamentals related to humanoid kinematics are presented in section 2.5, where a series of techniques to compute the forward and inverse kinematics are described, making a special accentuation on the prioritized scheme for solving the inverse kinematics procedure. This section also introduces the Jacobian matrix as an operator to compute the inverse kinematics in a differential modeling approach, explaining the different utilized versions of this matrix and the methods to construct them.

Section 2.6 presents all the aspects related to humanoid dynamics, presenting on one hand the concepts of Center of Mass (CoM), linear and angular momentums and depicting on the other hand the concept of Zero-Moment Point (ZMP), which is a key parameter to understand the nature of biped locomotion.

Finally, section 2.7 approaches a series of techniques to model the process of biped locomotion and to design walking pattern generators suitable for humanoid robots. This section also explains the fundamentals about the Preview Control of the ZMP, which is so far the most powerful tool to generate walking patterns which are dynamically stable.

2.1 Concept of humanoid robot

A robot is a unified dynamic system comprising electronics, software, and mechanical components. Furthermore, a humanoid robot is a robot with its overall appearance, based on that of the human body, allowing interaction with made-for-human tools or environments. In general humanoid robots have a torso with a head, two arms and two legs, although some forms of humanoid robots may model only part of the body, for example, from the waist up. Some humanoid robots may also have a face, with eyes and mouth. As a matter of fact, the most part of the science-fiction robots which can be seen in television or movies are provided with a human appearance, therefore, for many people, the humanoid robot is *the robot* by default.

According to the experts on the Humanoid Robotics field, in order to satisfy the most important functional requirements of functionality, they shall meet the characteristics illustrated in Figure 2.1.

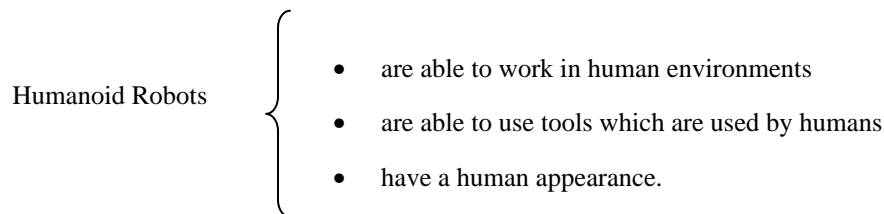


Figure 2.1 Characteristics of humanoid robots.

Considering the first characteristic, the physical environment of the modern society is conceived by the humans. For example, the lengths of a corridor, the height of a ceiling, or the position of a handrail are determined as a term of the size of the human body and the way humans move from one place to another. If the appearance of a robot, as well as the way it moves, is similar to those of a human, it is not necessary to modify the environment, since it could evolve. In contrast, in the case of wheeled mobile robots an irregular surface should be flattened, a narrow passage should be enlarged or an elevator should be needed to change altitude. This lack of need to modify the working environment represents an economic advantage when using humanoid robots.

The second characteristic is of the same order. Humans have designed most of the working tools they use for every activity. For example, the form and size of chairs and tables are determined so we can sit and eat using them. On the other hand, the geometry of the seat in a vehicle is designed to facilitate the access to the steering wheel, controls, and pedals. A humanoid robot should be able to use these same tools with such dexterity, at least, similar of a human's. Once again, the economic advantage is enormous, as it is not any more necessary to build specific instruments for humanoid robots.

The third characteristic requires certain explanation. It is so easy to personify a robot, as long as it keeps a human appearance. The more a robotic form gets far from a human form, the least humans will associate its behavior to that of a human. The human form is very important because the machine is perceived as a real companion, with which we could have communication capabilities. This is the reason that explains why every science fiction robot has a human appearance.

2.2 History of humanoid robots

Considering the history of real humanoid robots, a start point could be the WABOT-1, shown in Figure 2.2(a), and developed in 1973 by Ichiro Kato *et al.* in the University of Waseda [2] in Japan. It could be reasonable to consider WABOT-1 as *the first humanoid robot*. Even though the available technology at that time could make its conception as imperfect, it could walk, visually recognize objects and manipulate them with its two hands, it could also comprehend a spoken language and express itself using an artificial voice. The team of Ichiro Kato developed in 1984 WABOT-2, a humanoid robot which had the ability to play the piano as illustrated in Figure 2.2(b)



(a) WABOT-1



(b) WABOT-2

Figure 2.2 The humanoid robots of the University of Waseda [2]

But it was until 1996 due to the stunning revelation of humanoid robot P2 of Honda [3], that the time of humanoids really began. Honda launched a confidential humanoid robot project in 1986, just one year after the piano concert of WABOT-2. The robot P2 was 180 cm height and 210 Kg weight, and it was the first humanoid robot that could walk on its two legs in a stable way and with a totally built-in instrumentation. Honda presented its project P3 in 1997, which was 163 cm height and 130 Kg weight and three years later, it launched the humanoid robot ASIMO, which was 120 cm and 43 Kg. In Figure 2.3, a series of pictures of humanoid robot models produced by Honda are displayed, in a chronological order of production.

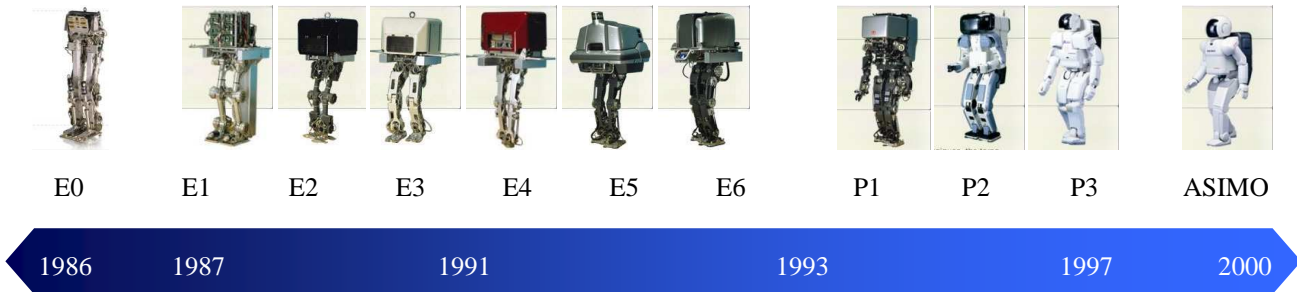


Figure 2.3 - Models of humanoid robots produced by Honda [3].

Before the public apparition of robot P2, most of the robotic community was being pessimistic about the development of a biped humanoid robot capable of walking in a stable way. Therefore the researchers felt amazed at the demonstration of Honda. The two most fundamental differences between the classic humanoid built until then and the P2 were, in first place, the difference of mechanism conception and, in second place, the adapted sensors.

The reduction systems in the known humanoid projects were constituted by rough gearings, with lots of mechanical backlash, while Honda's humanoid robot was designed with rigid and light mechanisms and used Harmonic Drive motors, without mechanical backlash and with a good torque capability.

On the other hand, the biped locomotion is unstable due to external perturbations, i.e. errors that exist between the theoretic model and the reality. As a result of this, the executed locomotion is unstable even if the theoretical walking model is stable. It is then necessary to adapt a feedback control loop, which allows obtaining information about the system status during its evolution inside the workspace. The utilization of sensors in the previous projects was not considered for the robot stabilization, while Honda provided its humanoid robots with high quality accelerometers and gyroscopes to determine linear and angular accelerations and also six axes force sensors to measure forces and torques at the contact points between feet and ground.

2.3 Constitution of humanoid robots

This section presents, from an introductory perspective, the constitution of humanoid robots, explaining on one hand the fundamentals of robot geometry, delimitating the different kind of bodies and joints that exist; and elucidating on the other hand the mechanical structure of a humanoid robot and its representation.

This section also approaches some of the most common hardware elements (i.e., sensors and actuators) that can be found in a humanoid robot platform and introduces as well the closed loop system used to control the articular movements on a robot.

2.3.1 Robot geometry, bodies and articulations

Mechanically, any robot is constituted by a series of bodies (also called links or segments), which are linked by articulations (or joints), which allow a relative movement between two consecutive bodies.

A body is a solid mechanical structure which connects two joints. The main purpose of a body is to maintain a fixed relationship between the joints at its ends. The last body of a kinematic chain has only one joint, located at the proximal end (the end closest to the base) of the body. At the distal end of this body (the end furthest away from the base) instead of a joint, there is usually a place to attach a working tool (e.g., a gripper, a welding gun or a laser device).

The common body configurations are shown in Figure 2.4. It can be seen that between the axes of the joints, at the ends of any body, there can be two degrees of translation and two degrees of rotation. These degrees of freedom are called the body parameters. The simplest body in (a) has two parallel revolute joints with no twist between the axes; the axes of the joints are parallel. If one of the joints in the previous body type is twisted about the centre line of the body by a certain angle, an extra degree of rotation is added. The twist angle is the angle that would exist between the joint axes if the joints were coincident, and it can be thought of as a rotation around the X axis. Thus, the second type of body in (b) has one degree of translation and effectively two degrees of rotation. In the type of body in (c), one joint of the first type of body is rotated 90° degrees so that the joint axis is collinear with the centre line of the body. The significant difference between this body and the previous two bodies is that the joint axes intersect, whereas in the previous two types of bodies the axes do not intersect.

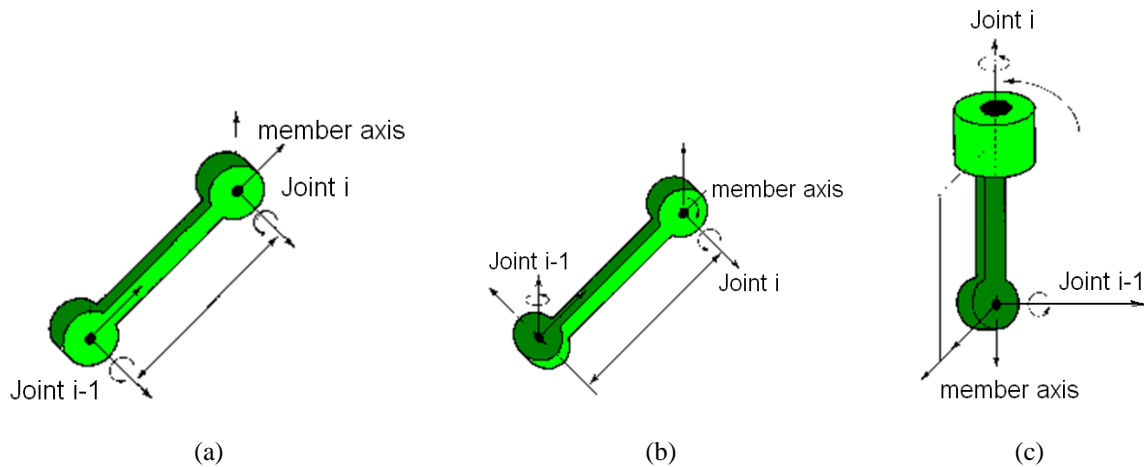


Figure 2.4 Types of bodies

There are two types of rotational joints, one where the axis of the joint is coincident with the centre line of the body (commonly used as a waist articulation), and the other where the axis of the joint is normal to the body axis (used as an elbow articulation), as shown in Figure 2.5(a).

On the other hand, a prismatic articulation is a sliding joint, with the axis of the joint coincident with the centre line of the sliding element. There are also two types of configurations, as shown in Figure 2.6: the one shown in (a) where the axis of the moving element is collinear to the preceding one and the other in (b) where the motion element axis is orthogonal to the previous body axis.

Commonly, the articulations used in humanoid robots are mainly rotational, since locomotion in humanoid robots tries to resemble as much as possible the human body motion. Prismatic articulations are more used for industrial manipulators and Cartesian robots.

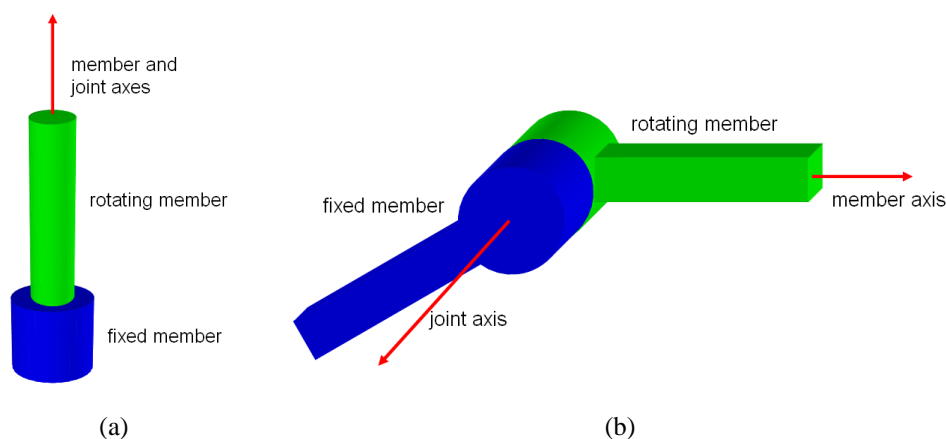


Figure 2.5 - Types of rotational articulations

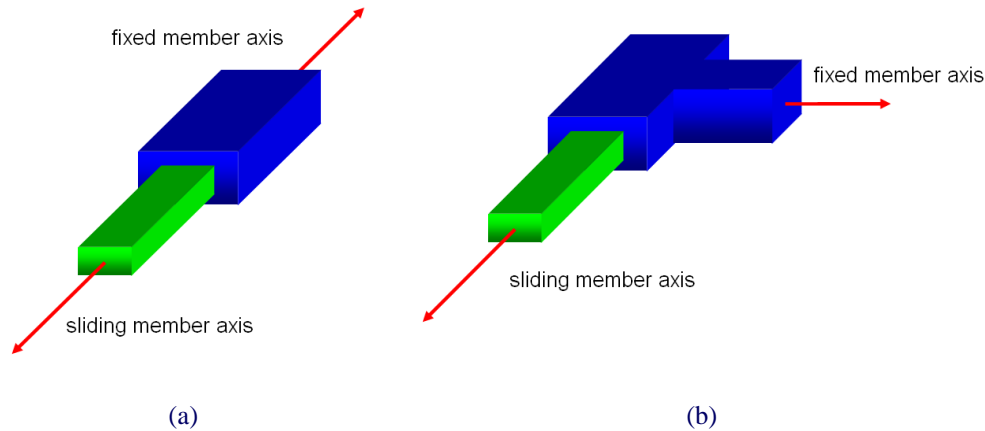


Figure 2.6 - Types of prismatic articulations

2.3.2 Mechanical structure of a humanoid robot and its representation

In particular, humanoid robots are composed by several kinematic chains of elements arranged in a tree-like structure. A **structural representation** of a humanoid robot is a decomposition of the system into a series of body bodies, linked to a series of articulations. There are two basic examples of representations, which are illustrated in Figure 2.7. The difference between them resides in the way to associate the bodies and the bodies. In (a) the articulations connected to the torso are linked to their bodies, and therefore there does not exist any body attached to the torso and all the other bodies are linked to only one articulation. This representation is called *two types of segments* representation, as there are only two body possibilities, torso or limb. In the case of (b) the articulations which are the closest to the torso are at the same time linked to it and in consequence, the amount of articulations linked to a body differs according to the considered body. In this configuration there exist *three types of segments*, according to the number of bodies attached to each body. Due to its simplicity, the first representation is the most commonly used for practical purposes.

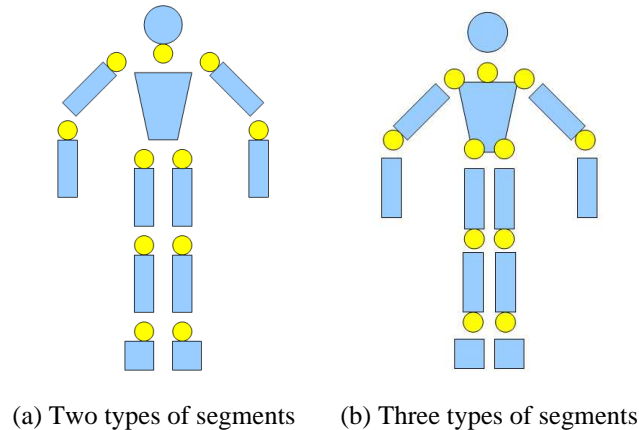


Figure 2.7 - Decomposition of a humanoid robot into segments.

Taking all bodies together to one adapted combination, it is possible to obtain a composition forming a humanoid robot. A common hierarchical structure in humanoid robots is the **tree-like structure**, presented in Figure. One could associate this structure to a genealogic tree, with each descendant having a common ancestor. The notations “R” (for right) and “L” (for left) are commonly used as prefixes to distinguish a considered side of the humanoid robot.

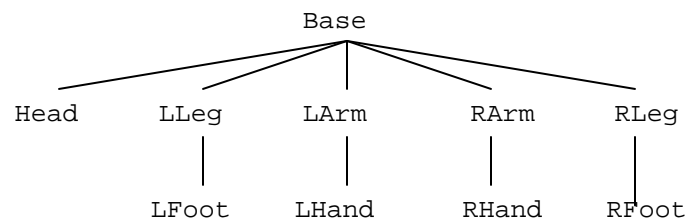
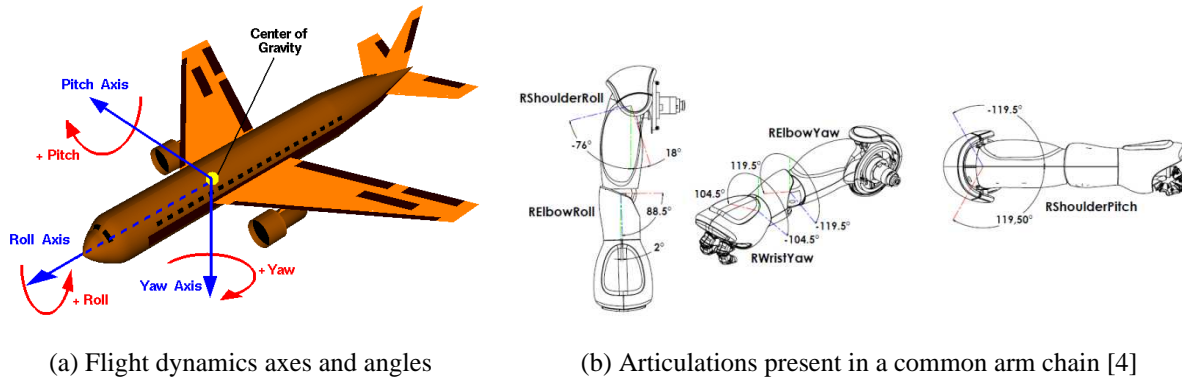


Figure 2.8 - Tree-like representation of the connections for a humanoid robot.

Another important concept is the term **end effector**. In robotics, an end effector is the device at the end of a robotic arm, designed to interact with the environment. The exact nature of this device depends on the application of the robot. In the strict definition, which originates from serial robotic manipulators, the end effector means the last link (or end) of the robot. At this endpoint the tools are attached. In a wider sense, an end effector can be seen as the part of a robot that interacts with the work environment. In the case of humanoid robots, the end effectors are commonly both feet, both hands and the head.

As it also occurs in industrial manipulators, it is common to make an analogy between the rotation angles that exist in humanoid robot articulations to those present in flight dynamics, as shown in Figure 2.9(a). Thus, the terms Roll, Pitch and Yaw are also used to differentiate one joint from the other inside one kinematic chain, as exemplified in figure Figure 2.9(b).



(a) Flight dynamics axes and angles

(b) Articulations present in a common arm chain [4]

Figure 2.9 - Analogy between humanoid articulations and the axes in flight dynamics

The amount of links and articulations present in a humanoid robot varies from one platform to another; nevertheless commonly, most humanoid robots are provided with six articulations for each kinematic chain; thus, for instance, if a humanoid has two legs and two arms it shall be equipped with 24 articulations. Additional articulations may be added for desired motion of the head, hands or fingers.

2.3.3 Actuators, movement transmissions and reducers used in humanoid robots

Actuators are the motors responsible for motion in the humanoid robot. Actuation of robotic systems is still an open question and represents a big challenge. Demanding performances including high power to mass ratio, capability of producing high power at low speed within a small-occupied volume are some of the key issues that required careful consideration. These criteria aimed to increase autonomy of humanoid robots.

Humanoid robots are constructed in such a way that they, from a certain perspective, imitate the human body, so they use actuators that perform like muscles and joints, though with a different structure. As previously mentioned, in order to achieve the same effect as human motion, humanoid robots use mainly rotary actuators. They can be electric, pneumatic, hydraulic, piezoelectric or ultrasonic.

Hydraulic and electric actuators have a very rigid behavior and can only be made to act in a compliant manner through the use of relatively complex feedback control strategies. While electric coreless motor actuators are better suited for high speed and low load applications, hydraulic ones operate well at low speed and high load applications. Piezoelectric actuators generate a small movement with a high force capability when voltage is applied. They can be used for ultra-precise positioning and for generating and handling high forces or pressures in static or dynamic situations. Ultrasonic actuators are designed to produce micrometric movements at ultrasonic frequencies (over 20 KHz).

They are useful for controlling vibration, positioning applications and quick switching. Pneumatic actuators operate on the basis of gas compressibility. As they are inflated, they expand along the axis, and as they deflate, they contract. If one end is fixed, the other will move in a linear trajectory. These actuators are intended for low speed and low/medium load applications. Between pneumatic actuators there are: cylinders, bellows, pneumatic engines, pneumatic stepper motors and pneumatic artificial muscles.

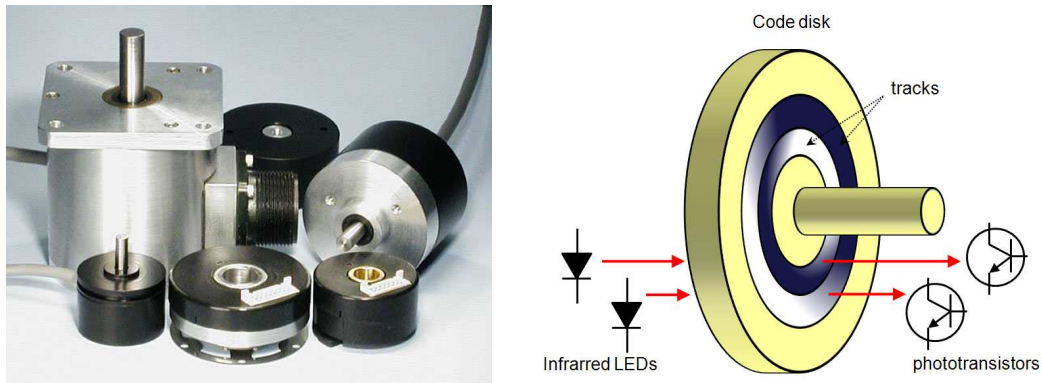
Transmissions are the devices which transmit movement from the actuators to the articulations. Since a robot moves the endpoint of its kinematic chains with an elevated acceleration, it is necessary to reduce the mass moment of inertia to the minimum. Therefore, the robot actuators are the closest to the robot base as possible and, as a consequence of this, robots use transmission systems which transport rotational movement from the actuators to the articulations. An efficient transmission system must have low size and weight, should avoid mechanic backlash, must not affect the movement it transmits and should be able to work continuously with an elevated torque, even during prolonged amounts of time.

2.3.4 Sensors used in humanoid robots

In order to perform a task with a desired accuracy, precision, velocity and intelligence, a robot should be equipped with sensors which will collect information about its own status (internal sensors) and about the environmental status (external sensors).

In the case of **internal sensors**, perhaps the most important ones are the **joint position sensors**, which measure the angular displacement of each articulation present in the kinematical structure. The most commonly sensors used for this purpose are the encoder and the resolver.

A digital optical encoder is a device that converts motion into a sequence of digital pulses. By counting a single bit or by decoding a set of bits, the pulses can be converted to relative or absolute position measurements. Encoders have both linear and rotary configurations, but the most common type is rotary. Rotary encoders are manufactured in two basic forms: the absolute encoder where a unique digital word corresponds to each rotational position of the shaft, and the incremental encoder, which produces digital pulses as the shaft rotates, allowing measurement of relative position of shaft. Most rotary encoders are composed of a glass or plastic code disk with a photographically deposited radial pattern organized in tracks. As radial lines in each track interrupt the beam between a photo emitter-detector pair, digital pulses are produced, as shown in Figure 2.10.

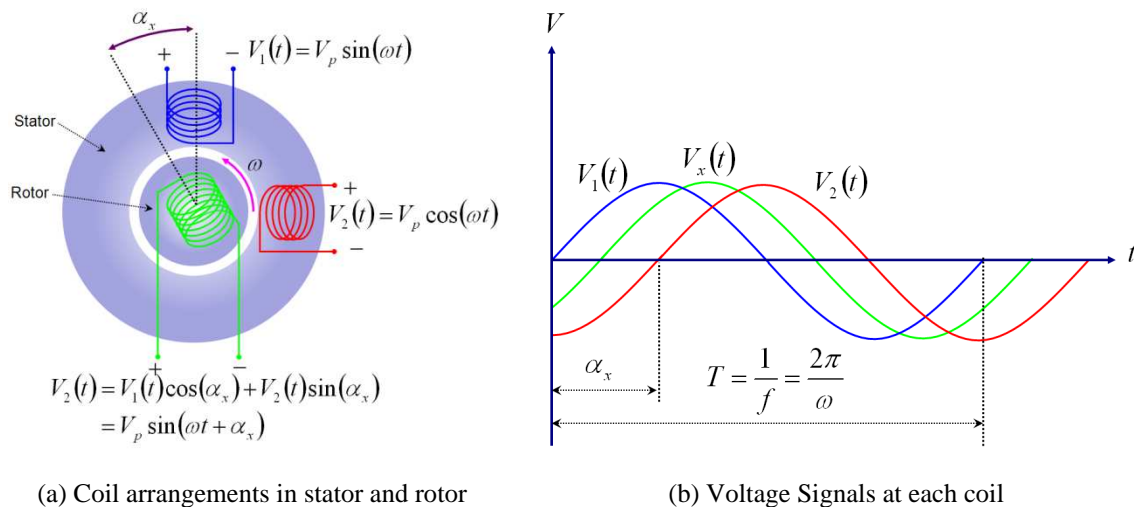


(a) Commercially available optical encoders (b) Operation principle of an incremental encoder

Figure 2.10 - Optical encoders used to measure joint angular position

A resolver is device used to convert angular position to an analog signal. It is based in an induction principle, where two fixed coils, located in a perpendicular geometrical arrangement on the stator are energized with a high frequency AC signal (normally between 5 and 20 KHz) and, as a result of this, a voltage is induced on a third coil which is located on the rotor, whose phase shift depends linearly on the position of the rotor shaft, as illustrated in Figure 2.11. Commonly, resolvers are designed to work at an operational speed up to 6000 rpm.

Another important internal variable which is needed to be measured in humanoid robots is the **angular velocity**. The information about movement velocity of each actuator is normally measured using a tachogenerator, which is a device that provides a d.c. voltage that is proportional to the motor shaft velocity.



(a) Coil arrangements in stator and rotor

(b) Voltage Signals at each coil

Figure 2.11 - Operation principle of a resolver

There are also other instances of internal sensors that used to measure the robot movements, respect to an inertial reference frame. These sensors are denominated **inertial sensors** and are basically the accelerometer and the gyroscope.

Accelerometers are sensors for measuring acceleration. Nonetheless, accelerometers actually measure the force exerted on mass since acceleration cannot be measured directly. The basic physical principle behind accelerometers is that of a simple mass spring system, as illustrated in Figure 2.12.

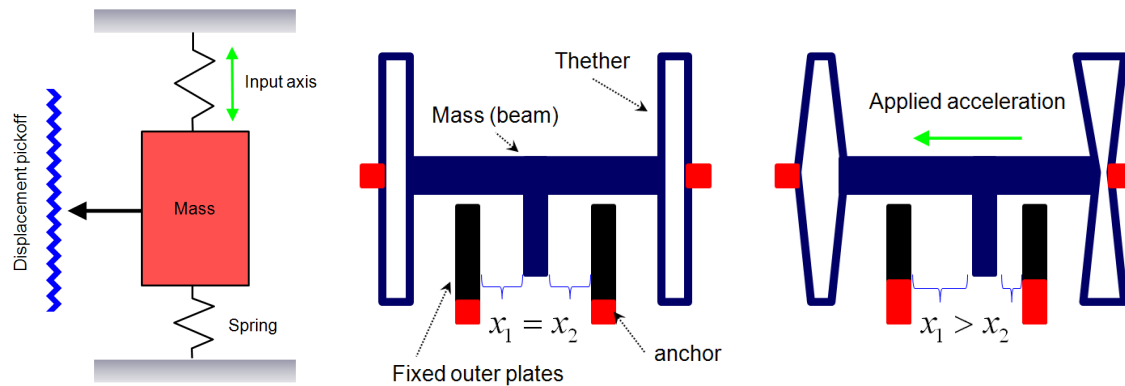


Figure 2.12 – Principle of operation of an accelerometer

Gyroscopes are used to measure angular rate; i.e., how quickly an object turns. The rotation is typically measured in reference to one of three axes (x , y or z). Depending on the way a gyroscope is placed on the reference frame, its primary axis of sensitivity can be one of these three axes of motion.

The principle of operation of these devices is based on the Coriolis Effect that is present when an object exhibits angular motion, they also are provided with a set of mass-spring systems as it is shown in Figure 2.13.

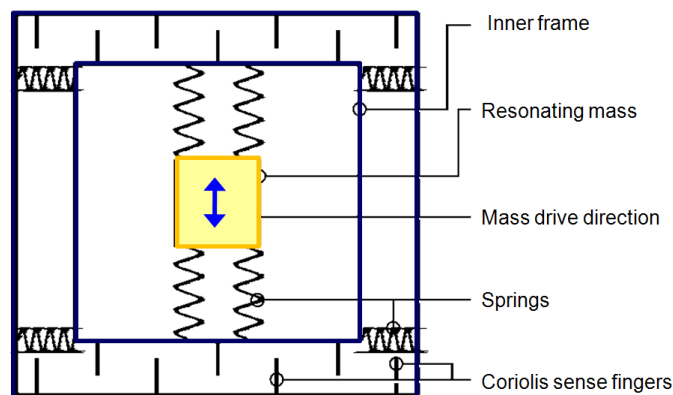


Figure 2.13 – Principle of operation of a gyroscope

By its part the second type of these transducers used in robotic platforms corresponds to the **external sensors**. These sensors get information about the environment and use it for specific purposes.

Particularly, some humanoid robots are provided with **force sensors** on the feet soles which used to measure the real position of the ZMP, as it will be later explained in section 2.6.3. The sensors used for this purpose are basically stress sensors of six axes, whose constitution is organized to simultaneously measure the external force $f = (f_x \ f_y \ f_z)^T$ and the external torque $\tau = (\tau_x \ \tau_y \ \tau_z)^T$ that are applied to a robot end effector. Some examples of these kinds of sensors are shown in Figure 2.14.



Figure 2.14 – Commercial stress sensors of six axes [5]

Moreover, some humanoid robots are also provided with image acquisition devices, in order to perform visually controlled tasks, like self-location, object recognition, map outlining, visual guidance, visual servoing or landmark pointing. Besides, audio sensors are also typical of some humanoid robots, which are intended to perform voice pattern recognition or audio recording.

2.3.5 Control system of a humanoid robot

The information collected from the internal sensors is taking inside a servo control loop, as shown in Figure 2.15, in order to control the motion of the actuators. This is the most common way to control the movement of a humanoid robot, and generally any kind of manipulator robot. Each articulation is then provided with a servo control loop like this.

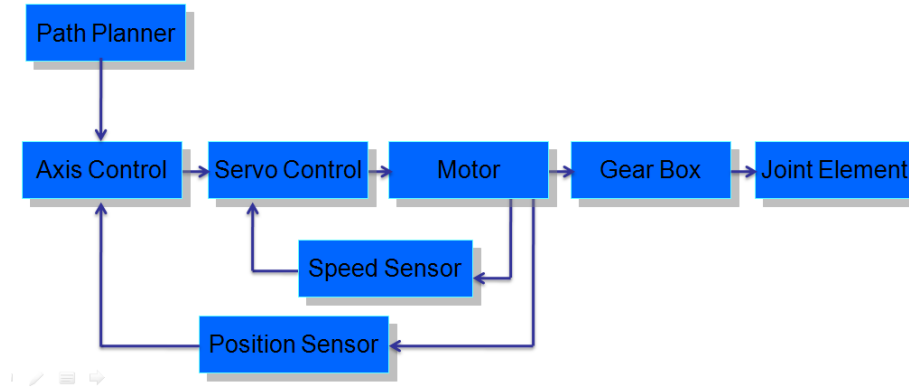


Figure 2.15 - Servo control loop of one robot joint

As illustrated, a path planner defines each joint angular position as a function of time, describing a chain endpoint trajectory; this reference information is given as an input to each joint servo control loop so each articulation is driven to the desired position and at the desired time and consequently, be able to perform a particular motion pattern, like walking, crouching, standing, grasping and so on.

2.4 Important aspects in humanoid robots

Beside the quite evident characteristics shown in Figure 2.1, humanoids can also be identified from other robotic platforms because they exhibit three important features which are redundancy, underactuation and non-holonomy.

As previously stated, the consideration of these characteristics is essential when performing the kinematic and dynamic modeling, as well as the locomotion planning. This section describes the concept of these important aspects.

2.4.1 Degrees of freedom in humanoid robots and concept of redundancy

In any type of robot, each independent movement that an articulation is able to perform in relationship with the previous one is defined as **degree of freedom (DOF)**. Furthermore, the number of DOF of a robot is given by the sum of the DOF of each articulation which constitute its mechanical structure.

As previously mentioned, the type of articulations which are used in humanoid robots are the rotational joints which, unlike the articulations that are present in the human body, are restricted only to one DOF, in order to simplify the mechanics, kinematics and control.

In order to locate and orientate a body in the three-dimensional space, six parameters are needed, as shown in Figure 2.16. Three of these parameters define its position and the other three, its orientation. This means that six DOF are required to perform this task. When more than six DOF are used in order to locate and orientate a robot in the space, it is said that there exist **redundancy** in the robot constitution, as happens in the case of humanoid robots, which are inherently provided with more than six DOF.

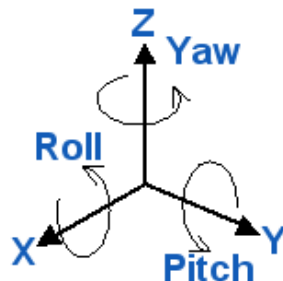


Figure 2.16 - Six DOF are required to define the position and orientation of a body in the space

2.4.2 Humanoids as mobile robots and concept of non-holonomy

According to [6], some studies on the steering of human locomotion at the trajectory planning level suggest that, for a pair of configurations (x, y, θ) and in the absence of obstacle, humans choose a common walking pattern among an infinite number of solutions. Moreover, it has been validated experimentally that a differential system satisfying the well known **non-holonomic** constraint arising in several wheeled mobile robots is a good approximation of human walking. The non-holonomic behavior of human walking is activated when the trace of the body direction corresponds to the tangent direction of the path. The differential coupling between the body position and direction is given by

$$\dot{y} \cos \theta - \dot{x} \sin \theta = 0 \quad (2.1)$$

In other words, the previous formulation means that the robot orientation will always tend to coincide with the tangent line at the current point of the displacement trajectory on the plane; however, other human locomotion strategies are also valid, e.g., backward and sideward steps. The switching decision between locomotion strategies may not be based only on the distance from the initial to final configurations but also for obstacle avoidance.

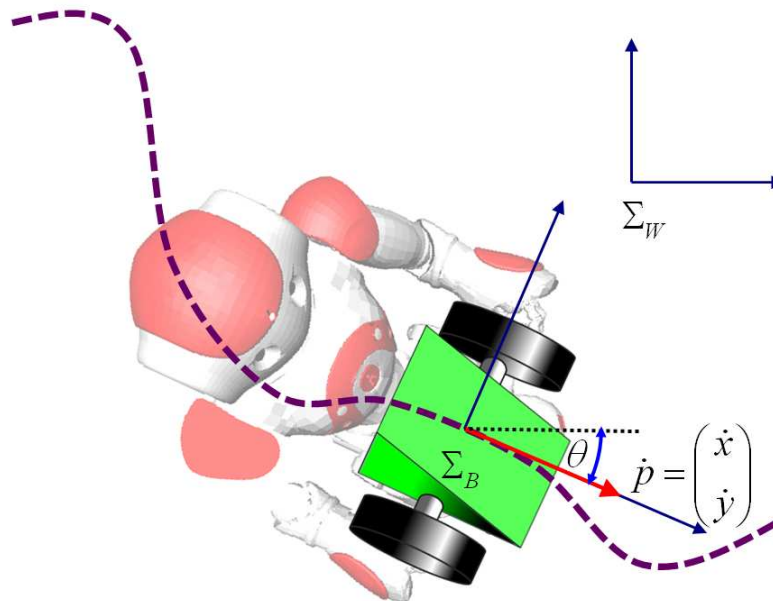


Figure 2.17 – Humanoid modeled as a differential-drive robot and concept of non-holonomy

2.4.3 Underactuated nature of humanoid robots

According to Newton's second law, the dynamics of mechanical systems are second order ($F = m \cdot a$). Thus, their state is given by a vector of positions, q , and a vector of velocities, \dot{q} , and possibly, time. The general form for a second-order controllable dynamical system is:

$$\ddot{q} = f(q, \dot{q}, u, t) \quad (2.2)$$

where u is the control vector. The forward dynamics for many of the robots that we care about turn out to be affine in commanded torque, so let's consider a slightly constrained form:

$$\ddot{q} = f_1(q, \dot{q}, t) + f_2(q, \dot{q}, t)u \quad (2.3)$$

A control system described by equation (2.3) is considered to be fully actuated in configuration (q, \dot{q}, t) if it is able to command an instantaneous acceleration in an arbitrary direction in q ; otherwise it is considered to be **underactuated**.

Whether or not a system is underactuated may depend on the state of the system. In other words, underactuated control systems are those in which the control input cannot accelerate the state of the robot in arbitrary directions. As a consequence, unlike fully-actuated systems, underactuated systems cannot be commanded to follow arbitrary trajectories.

In particular, a humanoid robot is inherently underactuated due to the mobile nature of its base. As it will be explained in section 2.5.5, modeling a mobile manipulator implies the consideration of six additional DoF, describing the base position and orientation. Thus, a humanoid robot, modeled as a mobile manipulator, is constituted by a total amount of $n+6$ DoF where n is the total number of articulations in the model.

The DoF that are associated to an articulation of the kinematic model are considered as the actuated DoF of the model, since their role depends on an actuator which governs the articulation movement. In contrast, the mobile base additional DoF are not associated to any kind of actuator; therefore these DoF are considered to be not-actuated. Finally, the presence of actuated and not-actuated DoF in the kinematic model is the foundation which makes a humanoid robot an underactuated system.

2.5 Kinematics of Humanoid Robots

The theory that analyzes the relationship between the location (position and orientation) of the robot bodies and the articular configuration is called **kinematics**. It is the base for the description of the robotic motion and the graphic representation of robot structures. Particularly, the kinematics stage is the first main tool considered in the design of a complete walking pattern generator. This section presents the kinematic tools that are necessary to solve humanoid locomotion problems, considering the aspects of redundancy and non-inertial nature, explained in the previous section and that are characteristic of this type of robots.

The first part of this section describes the way a kinematic structure is modeled, considering a mobile base. The second part of this section presents the forward kinematics procedure used to determine the location (i.e., position and orientation) of any point related to the kinematic model from a given articular configuration. Subsequently, this section explains the direct geometric model time differentiation and the way to construct the associated Jacobian matrices and finally, some procedures to determine the articular configuration that relocates a set of selected points (related to the kinematic structure) to specific locations or regions, considering different levels of priority.

The complete definition of the movement of an object in the space needs a series of algorithms that are founded on mathematic background; all of these are correspondingly approached through the following subsections.

2.5.1 Coordinate system frames

The first step in the kinematics stage is to define a point in the space and then, regarding this point as the origin, it is possible to define a fixed **world reference frame** Σ_w by considering the axis x as the direction pointing towards, y to the left and z to above. All parameters (e.g., position, orientation, velocity) expressed on this frame will be considered as *world* or *absolute* parameters.

Furthermore, it is possible to define an N number **local frames** Σ_j , with $j = 1, 2, \dots, N$, each of them associated to a rigid body j . These local frames are used to express a determined parameter, respect to the body j , regarding this parameter as *local* or *relative*.

In contrast to the world reference frame, which stays always fixed in space, a local frame can move, attached to the body it is associated to. Figure 2.18 shows an example of world and local frames.

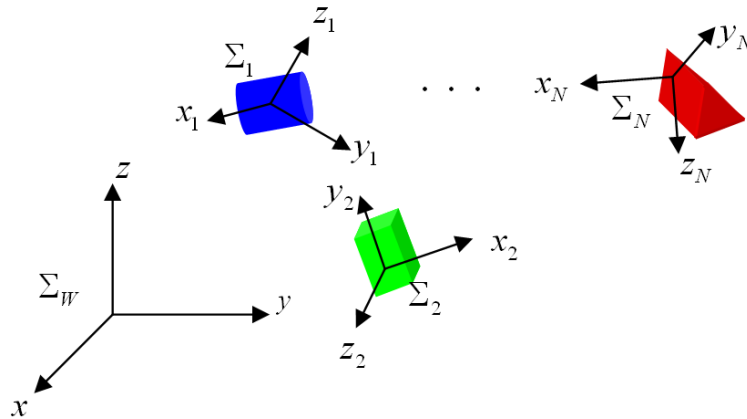


Figure 2.18 - World and local coordinate system frames

2.5.2 Generation of the kinematic model

A fundamental part of kinematics is the construction of an accurate model of the robot. To do this, the designer should select a convenient **structural representation** for the model, based on those presented in section 2.3.2. As previously mentioned the *two types of segments* representation is the most suitable for practical purposes, since it allows an easier recursive programming. In this representation, the model is constituted by one base body and a finite number of bodies that are associated each of them to a specific articulation.

Now, by defining an identification number (commonly equal to one) for the base body and subsequent positive integers for the rest of the bodies, it is possible to establish a connection among all bodies that constitute the kinematic model by the definition of a unique **parental relationship** *mother*. This connection implies that any individual body j of the model, except the base, is physically associated via an articulation to one (and only one) mother body i , generating a successive kinematic chain towards the model base. A simple example of this procedure is illustrated in Figure 2.19, showing a graphic representation of a tree-like structure with five DOF and establishing the parental relationship among the model bodies.

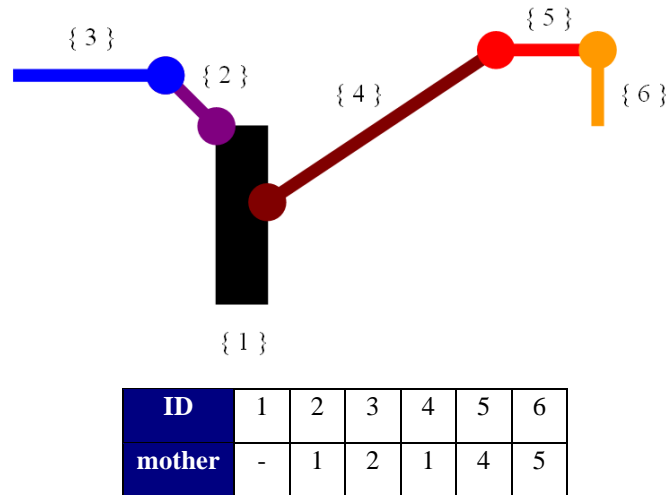


Figure 2.19 - Example of a tree-like structure with five DOF and two kinematic chains. The table below the illustration shows the parental relationship among the model bodies. The base has been identified with number 1.

The next step in the creation of the robot model is to define the **position and orientation of the local coordinate frames** (Σ_j) for each model body j . A recommendable (but not unique) method to attain this is explained below and exemplified in Figure 2.20.

- The origins of the local frames should be placed at a point on the respective joint rotation axis. It is common to locate these frame origins at the same point where both related bodies' axes intersect each other.
- The orientation of the local frames are defined in such form that all frames orientations coincide with the world reference frame (Σ_w) when the model is in the initial configuration (i.e., when all articular values q_j are set to zero).

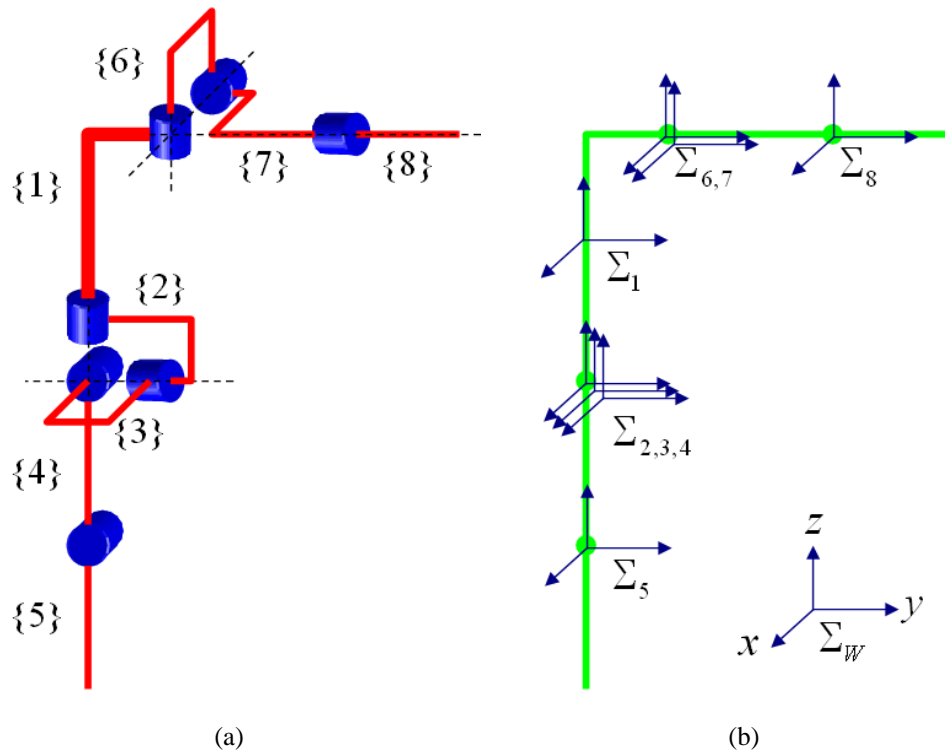


Figure 2.20 - (a) Kinematic structure in initial configuration of a robot model constituted by eight bodies and seven articulations. The bodies' identifiers are closed by brackets. (b) Definition of the position and initial orientation of the bodies local frames.

The following stage for the creation of the kinematic model is the definition of the **characteristic vectors** a_j and b_j , representing the rotation axes associated to each articulation, as well as the relative position, respectively. Both vectors must be defined for each body j , excluding the model base, as follows:

- The vectors a_j are unitary vectors that precise the direction of the joint rotation; they are expressed respect to the considered body parent frame and defined using the right hand rule (positive joint rotation is in direct trigonometric sense).
- The relative position vector b_j indicates the position of a local frame, respect to the parent frame. This vector is null if both frames are superimposed.

Figure 2.21 graphically shows the definition of the characteristic vectors a_j and b_j for the model example of eight bodies that was stated in Figure 2.20.

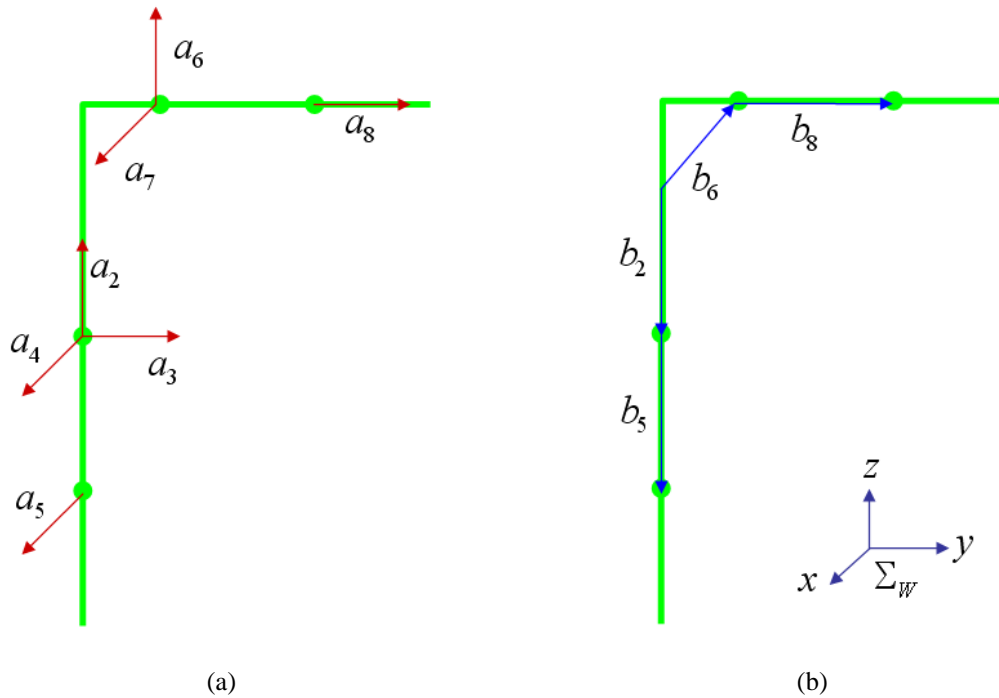


Figure 2.21 - Characteristic vectors for the kinematic structure presented in the previous example. (a) Axes rotation vectors a_j . (b) Relative position vectors b_j

The representation that was just described above is used in all algorithms needed to compute the forward kinematics, differential kinematics, prioritized inverse kinematics and finally, the ZMP dynamics, all of them used to generate the walking pattern in humanoid locomotion planning, which is the main objective of this master thesis project.

It is convenient to mention that there also exists a very well known method to describe the relative positions and orientation of the local frames, denominated *Denavit – Hartenberg Convention* (DH). This method presents a restriction in which the modeler is obligated to change the orientation of a rotation axis when changing from one body to another. This characteristic propitiates the apparition of errors and therefore the DH representation is not utilized in this work.

2.5.3 Representation of the position and orientation in the space of a rigid body

In the attempt to control the motion of a robot, it is fundamental to be able to precisely define the position and orientation of each body of its kinematic structure. In general terms, in order to represent the position and the orientation of a rigid body A , related to the world reference frame Σ_W in \mathcal{R}^N , a series of scalar values are required. Some of

them define the body position and other define the body orientation. All transformations that are applied to the body A are performed inside the **Euclidean Special Group**, $SE(N)$. This group is composed by a set of translations $p_A \in \mathfrak{R}^N$ and a set of rotations R_A that are part of the **Orthogonal Special Group** $SO(N)$, which is also part of the set of real and non-singular matrices in the **General Linear Group** $GL(N)$.

Particularly, when $N = 3$, a total amount of twelve parameters are needed to represent the body location, where six describe the position and the other nine, the orientation. Thus, an element of $SE(3)$ has the following structure:

$$T_A = \begin{bmatrix} R_A & p_A \\ 0 & 1 \end{bmatrix} \in SE(3) \quad (2.4)$$

where $R_A \in SO(3)$ and $p_A \in \mathfrak{R}^3$

The rotation of a free body A in the space is determined by using the rotation of the body frame Σ_A , respect to the world reference frame Σ_W , as illustrated in Figure 2.22

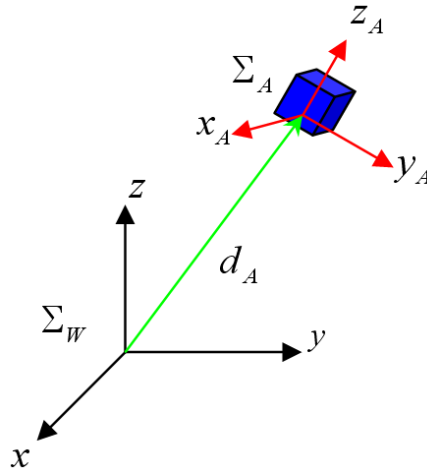


Figure 2.22 - Rotation of a body A in the space

All possible rotations of Σ_A respect to Σ_W can be represented by using a **rotation matrix** R_A , which describes the orientation of the body A and quantifies its rotation. In general terms, any rotation matrix R in $SO(3)$ has the following structure

$$R = [R_1 \quad R_2 \quad R_3] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{where } r_{ij} \text{ is a scalar}$$

This Matrix R is **orthogonal** and therefore fits the following algebraic constraints:

$$R_1^T R_1 = 1, R_2^T R_2 = 1, R_3^T R_3 = 1 \quad (2.5)$$

$$R_1^T R_2 = 0, R_2^T R_3 = 0, R_3^T R_1 = 0 \quad (2.6)$$

Furthermore, it is possible to establish a minimum amount of independent parameters that are needed to describe the position and orientation of a rigid body and it is given by

$$n = \frac{1}{2}N(N+1) \quad (2.7)$$

As a result of this, when $N = 3$, then $n = 6$. This means that the twelve parameters that are originally needed to describe the body position and orientation can be reduced to six elements, where three of them are used to represent position and the other three to represent orientation, considering a determined Euler angles convention (see Appendix A). This minimum set of parameters is known as **degrees of freedom (DOF)** of the mechanism.

As approached in section 2.3.2, there are three elementary rotations around the axes x , y and z that allow to describe the rotation of a body in the space and are denominated **roll**, **pitch** and **yaw**, respectively. The commonly used notations for these rotations are resumed in the following table:

Rotation axis	Name	Notation angle
x	Roll	ϕ
y	Pitch	θ
z	Yaw	ψ

Now, in order to represent the roll, pitch and yaw of an object in terms of a given angle, three different rotation matrices are used, each of them describing one elementary rotation, as described as follows:

$$\begin{aligned}
R_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \\
R_y(\theta) &= \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \\
R_z(\psi) &= \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.8}$$

The complete rotation in the space of a rigid body can be represented as the product of the three elementary rotation matrices $R_x(\phi)$, $R_y(\theta)$ and $R_z(\psi)$, ordered in an adequate permutation, according to a determined Euler angles convention. For instance, if the previously regarded rigid body A is known to have a known roll (ϕ_A), pitch (θ_A) and yaw (ψ_A), the corresponding rotation matrix $R_A \in SO(3)$ can be calculated by the relation

$$R_A = R_{z\psi x}(\psi_A, \theta_A, \phi_A) = R_z(\psi_A)R_y(\theta_A)R_x(\phi_A) \tag{2.9}$$

The matrix $R_{z\psi x}(\psi, \theta, \phi)$ groups the three elementary rotations and represents the whole rotation in the tridimensional space. This representation is denominated *roll-pitch-yaw* notation and corresponds to the z - y - x notation of Euler angles. Other notations might be also used instead of this one; nevertheless the one presented in this section is the most frequently used in navigation, aeronautics, and robotics, due to its simplicity and intuitive nature. In the subsequent sections, the z - y - x angles notation will be used to describe orientations, unless other statement is specified.

2.5.4 Forward Kinematics

The term *forward kinematics* makes reference to the method which allows the calculation of the position and orientation of the model bodies, as a function of given values for the articular configuration. Furthermore, the execution of this stage can be applied to determine other important parameters for the robot motion, such as the position of the CoM, the constitution of the support polygon, the distances among different bodies in order to avoid collisions, the orientation deviation to a target landmark or the location of any arbitrary point associated to a determined model body. The forward kinematics is the basis for all robotic simulation.

The forward kinematics procedure performs a linear transformation from the model Configuration Space (CS) described by the vector of generalized coordinates \mathbf{q} (representing the articular values of the model and has dimension n), to the model Cartesian Operational Space (OS) described by a pose vector \mathbf{X} (constituted by three parameters for position and three for rotation, considering a selected notation of Euler angles). Intuitively, this linear transformation is performed in the way illustrated in Figure 2.23.

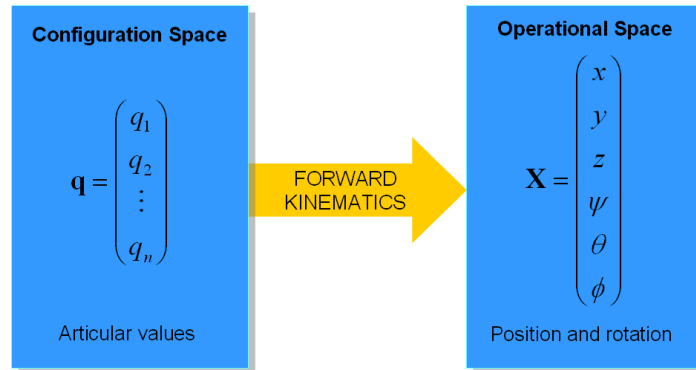


Figure 2.23 - Forward Kinematics maps from the Configuration Space (CS) to the Cartesian Operational Space (OS)

Thus, the Operational Space (OS) represents all locations (positions and orientations) that are reachable by a considered model body, as a function of all possible sets of articular values in the Configuration Space (CS).

Absolute Position and Orientation

The model forward kinematics can be calculated by using a series of homogeneous transformations. Consider a rigid body j , as the one shown in **¡Error! No se encuentra el origen de la referencia.**, with the following known elements:

- Local frame Σ_j whose origin is placed over the body j rotation axis.
- Local frame Σ_i , associated to the parent of body j .
- Joint rotation axis vector a_j , expressed respect to the parent frame Σ_i .
- Relative position vector b_j , expressed respect to the parent frame Σ_i .
- Articular variable q_j , representing the joint rotation in direct trigonometric sense.
- Absolute position vector p_i of the parent body i .
- Absolute orientation matrix R_i of the parent body i .

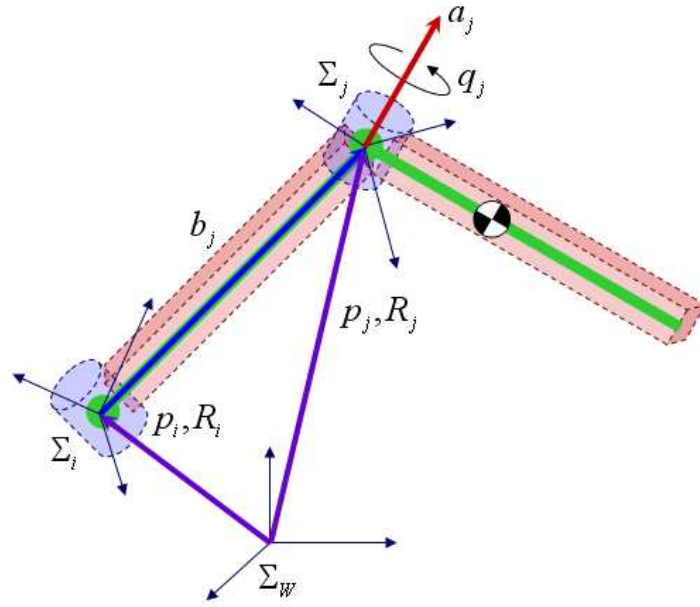


Figure 2.24 - Determination of the absolute position (p_j) and orientation (R_j) of a body j as a function of the articular value q_j . The characteristic vectors a_j and b_j represent respectively the axis rotation and the origin of the local frame Σ_j , respect to the parent frame Σ_i .

Having the information summarized above, it is possible to establish a homogeneous transformation T_j as follows:

$$T_j = \begin{bmatrix} R_i & p_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e^{\hat{a}_j q_j} & b_j \\ 0 & 1 \end{bmatrix} \quad (2.10)$$

The rotation matrix $e^{\hat{a}_j q_j}$ used in the previous expression is calculated in the following way with the knowledge of parameters a_j and q_j of body j , by using the *Equation of Rodrigues*, which is explained with detail in Appendix B.

$$e^{\hat{a}_j q_j} = I + \hat{a}_j \sin q_j + \hat{a}_j^2 (1 - \cos q_j) \quad (2.11)$$

where $\hat{a}_j \in \mathfrak{R}^{3 \times 3}$ is an anti-symmetric matrix obtained from the axis rotation vector a_j by using the following notation:

$$\hat{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \quad (2.12)$$

By using the homogeneous transformation established in equation (2.10) it is possible to define the **absolute position** (p_j) and **orientation** (R_j) of the local frame of body j , which are given by:

$$p_j = p_i + R_i b_j \quad (2.13)$$

$$R_j = R_i e^{\hat{a}_j q_j} \quad (2.14)$$

With both relations shown above and considering the structural representation previously described, it is possible to design a recurrent algorithm to determine the **absolute position and orientation of all bodies** constituting the complete kinematic model; it is only necessary to define the position and orientation of the base body. The Algorithm 2.1 presented in this section develops the forward kinematics by following the method described in this section.

Algorithm 2.1 Forward Kinematics to determine the position and orientation of all bodies that constitute the model structure

Input: Amount of bodies N and parental relationship among bodies; position vector p_1 and orientation matrix R_1 of the model base; axis rotation vector a_j , relative position vector b_j and articular variable q_j of all joint-associated bodies.

Output: Position vector p_j and orientation R_j of all joint-associated bodies.

- 1: **for** $j = 2$ to $j = N$ **do**
 - 2: Define i as the parent of j
 - 3: Calculate body absolute position $p_j = p_i + R_i b_j$
 - 4: Calculate body absolute orientation $R_j = R_i e^{\hat{a}_j q_j}$
 - 5: **end for**
-

Furthermore, when the absolute position and orientation of a considered body j has been calculated by the use of equations (2.13) and (2.14), it is also possible to determine the **absolute position** c_j of an arbitrary point that is associated to the body j , by the knowledge of its relative position \bar{c}_j respect to the local frame Σ_j , as illustrated in Figure 2.25, by using the following equation:

$$c_j = p_j + R_j \bar{c}_j \quad (2.15)$$

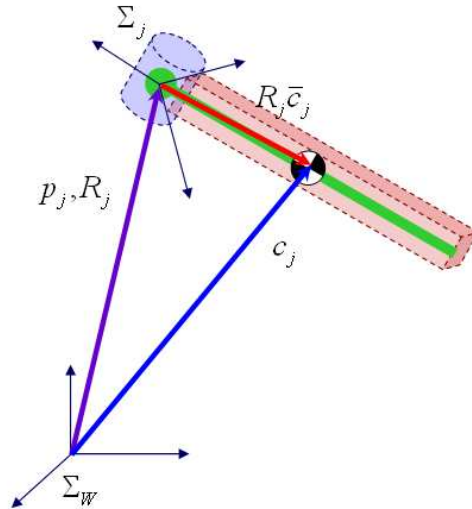


Figure 2.25 - Calculation of the absolute position c_j of an arbitrary point that is associated to the body j

Linear and Angular Velocities

There are some cases in which not only the knowledge of position and orientation of a model body is required, but also its linear and angular velocities. These parameters can be determined as a function of the following:

- Linear velocity of parent frame v_i .
- Angular velocity of parent frame ω_i .
- Joint rotation axis vector a_j , expressed respect to the parent frame Σ_i .
- Relative position vector b_j , expressed respect to the parent frame Σ_i .
- Articular velocity \dot{q}_j , in direct trigonometric sense.
- Absolute orientation matrix R_i of the parent body i .

Thus, the linear velocity of body j is given by:

$$v_j = v_i + \omega_i \times R_i b_j \quad (2.16)$$

and its angular velocity is determined as follows:

$$\omega_j = \omega_i + R_i a_j \dot{q}_j \quad (2.17)$$

Evidently, the utilization of the previous two equations implies that the linear and angular velocities of the model base are known.

An alternate way to calculate these parameters is by the knowledge of the variations in time of the absolute position vector (\dot{p}_j) and the orientation matrix (\dot{R}_j) of body j . The linear and angular velocities can be determined by the following relations:

$$v_j = \dot{p}_j \quad (2.18)$$

$$\omega_j = (\dot{R}_j R_j^T)^\vee \quad (2.19)$$

Where the operator $(\)^\vee$ represents the vector associated to an anti-symmetric matrix, in the reciprocal way as described in equation (2.12).

2.5.5 Differential kinematic modeling and the Jacobian matrix

The objective of this section is to introduce the Jacobian matrix as an element that relates the infinitesimal articular variations in the Configuration Space (CS) with their effects in the Cartesian Operation Space (OS). By using the Jacobian matrix, it is possible to have access to the necessary articular joint values to generate an external effort in the robot bodies, in order to locate them at a desired position and orientation. This method is frequently utilized in the control of robots and considers velocity-level reasoning.

There are several types of Jacobian matrices that are used in the control of robot motion. Each of these matrices has a determined function for practical purposes. In order to attain the objectives of this work, four different types of Jacobian matrices are approached:

- Geometric Jacobian matrix
- Analytical Jacobian matrix
- Mobile manipulator geometric Jacobian matrix
- Mobile manipulator analytical Jacobian matrix

This section describes the mathematical foundation about these matrices and also explains a method to construct them according to [7] and [8].

The Geometric Jacobian Matrix

Considering a model that is constituted by a set of articulated links, forming an open kinematic chain, the linear and angular velocity of one of its bodies (j) is determined by the variation of its position and orientation with respect to time. From this definition and having knowledge of the absolute position of this body by using equation (2.13), the linear velocity is defined as $\dot{p}_j \in \mathfrak{R}^3$ for a tridimensional space and is given by:

$$\dot{p}_j = \frac{\delta p_j}{\delta q} \frac{dq}{dt} = J_v(q) \dot{q} \quad (2.20)$$

In the previous equation $q \in \mathfrak{R}^n$ represents the articular values in the Configuration Space (CS), where n defines the amount of DoF of the kinematic model. Thus, the transformation that relates the body linear velocity \dot{p}_j with the articular velocity \dot{q} is determined by the following Jacobian matrix:

$$J_v(q) = \frac{\delta p_j}{\delta q} \in \mathfrak{R}^{3n} \quad (2.21)$$

Furthermore, in order to calculate the angular velocity of the body j , it is necessary to consider that this velocity depends on the sum of the partial velocities that are provided by each individual body which constitutes the kinematic chain from j towards the model base. This formulation can be generalized in the following way:

$$\omega_j = \omega_i + \rho_j Z_j \dot{q}_j \quad (2.22)$$

where ω_i represents the angular velocity of the parent body i , and the scalar ρ_j is equal to one if the articulation associated to body j is rotational or equal to zero if it is translational. The vector $Z_j \in \mathfrak{R}^3$ corresponds to joint rotation axis vector, expressed in absolute coordinates and is given by:

$$Z_j = R_i a_j \quad (2.23)$$

where, as explained in the previous section, R_i is the rotation matrix of the parent body i and a_j is the rotation axis vector of the joint associated to body j respect to the parent frame Σ_i . Based on the formulation of equation (2.22), the angular velocity of body j can also be expressed in a compact form, as follows:

$$\omega_j = J_\omega(q)\dot{q} \quad (2.24)$$

where $J_\omega(q) \in \mathfrak{R}^{3 \times n}$ is the Jacobian matrix that relates the angular velocity of a body with the articular velocity of the model.

Now, consider an arbitrary frame Σ_c that is associated to the body j and relatively fixed respect to the local frame Σ_j , as shown in Figure 2.26. Its relative position \bar{p}_{c_j} and its rotation deviation respect to R_j are always constant. As a result of this, the position and orientation of the frame Σ_c will be affected by the articular variations of all joints that are related to those bodies which constitute the kinematic chain from the body j till the model base. Any articulation associated to a body that is not part of this route will not affect the location of the frame Σ_c .

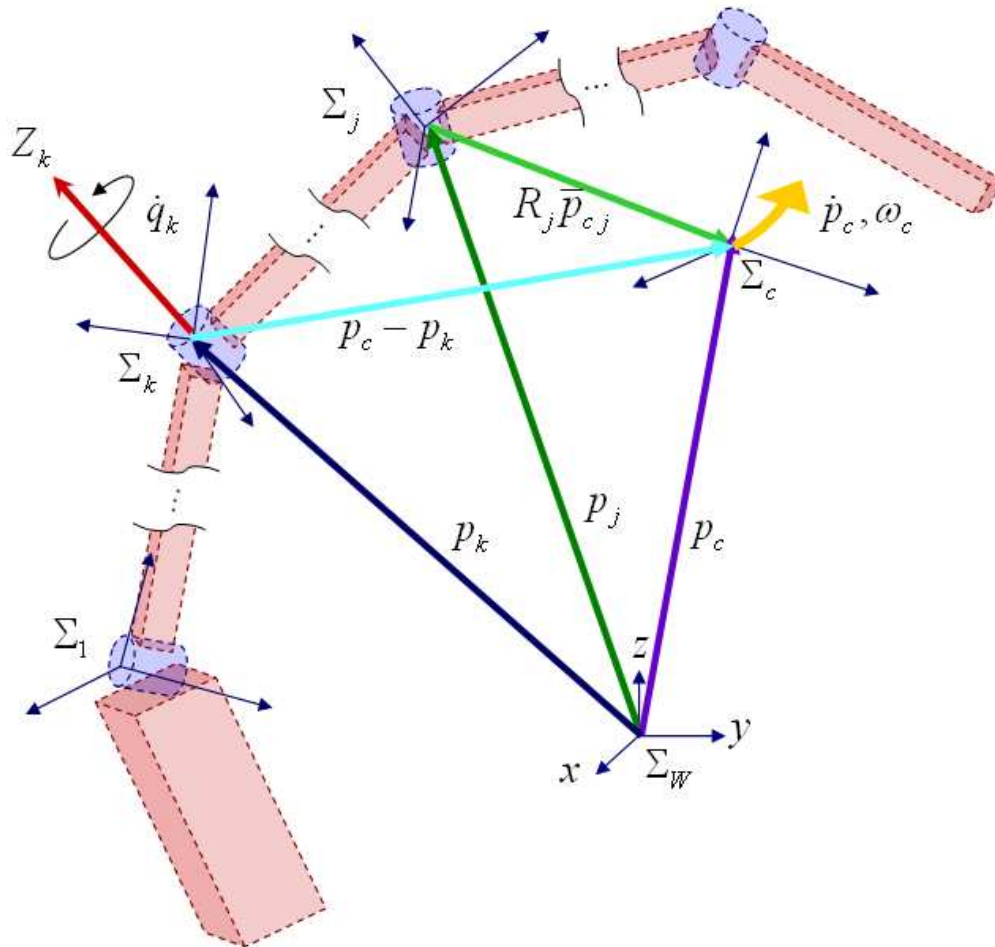


Figure 2.26 - Calculation of one column of the geometric Jacobian matrix: The movement of the arbitrary frame Σ_c , whose location is relatively fixed to frame Σ_j , results of the variations of the articular values of those bodies constituting the kinematic chain from body j to the model base.

By using a homogeneous transformation similar to that stated in equation (2.15), the position of the frame Σ_c , respect to world coordinates can be calculated by:

$$p_c = p_j + R_j \bar{p}_{c_j} \quad (2.25)$$

Furthermore, from the previous formulations, it is possible to construct a complete **Geometric Jacobian Matrix** $J_{G_c}(q) \in \mathfrak{R}^{6 \times n}$, which relates the linear and angular velocities of the frame Σ_c with the articular velocities of the model. This matrix is constituted by n columns, each of them associated with one articular variable of the Configuration Space (CS). Thus, as illustrated in Figure 2.26, if a considered body $k = \{2, 3, \dots, n\}$ is part of the kinematic chain from body j to the model base, the $(k-1)$ -th column of the linear part $J_{v_c}(q) \in \mathfrak{R}^{3 \times n}$ and angular part $J_{\omega_c}(q) \in \mathfrak{R}^{3 \times n}$ of the geometric Jacobian matrix associated with the movement of frame Σ_c as a function of the articular values are calculated, respectively, in the following way:

$$J_{v_{ck}}(q) = \begin{cases} Z_k \times (p_c - p_k) & \text{rotational articulation} \\ Z_k & \text{translational articulation} \end{cases} \quad (2.26)$$

$$J_{\omega_{ck}}(q) = \begin{cases} Z_k & \text{rotational articulation} \\ 0 & \text{translational articulation} \end{cases} \quad (2.27)$$

where $J_{v_{ck}}(q)$ and $J_{\omega_{ck}}(q)$ are expressed in \mathfrak{R}^3 . In the case that the considered body k is not part of the kinematic chain then $J_{v_{ck}}(q) = J_{\omega_{ck}}(q) = 0$.

Finally, the geometric Jacobian matrix $J_{G_c}(\dot{q})$ is obtained by the vertical concatenation of both linear and angular parts by:

$$J_{G_c}(q) = \begin{bmatrix} J_{v_c}(q) \\ J_{\omega_c}(q) \end{bmatrix} \in \mathfrak{R}^{6 \times n} \quad (2.28)$$

Thus, the vector $v \in \mathfrak{R}^6$ which describes the total velocity (linear and angular) of the frame Σ_c in terms of the articular variables given by:

$$v_c = \begin{bmatrix} \dot{p}_c \\ \omega_c \end{bmatrix} = J_{G_c}(q) \dot{q} \quad (2.29)$$

The Algorithm 2.2 shows a recursive procedure to compute the geometric Jacobian matrix of a frame Σ_c that is associated to a body of a kinematic model of N bodies.

Algorithm 2.2 Computation of the geometric Jacobian matrix

Input: Amount of model joint-bodies n and parental relationship among them; base-body and joint-bodies absolute position vectors and orientation matrices; particular body j which frame Σ_c is associated to; relative position vector \bar{p}_c of frame Σ_c , respect to Σ_j .

Output: Geometric Jacobian matrix $J_{G_c}(q)$.

- 1: Initialize geometric Jacobian to zeros; $J_{G_c}(q) = \mathbf{O}^{6 \times n}$
 - 2: Calculate the absolute position of frame Σ_c by using (2.25)
 - 2: Determine the bodies that constitute the kinematic route to from body j to the model base
 - 3: **for** all element h in the kinematic route **do**
 - 4: Define k as the ID of the body corresponding to the h -th element of the route
 - 5: Calculate absolute rotation axis vector $Z_k = R_k a_k$
 - 6: Rewrite column $k-1$ of the geometric Jacobian matrix $J_{G_c}(q)$ by using equations (2.26), (2.27) and (2.28).
 - 7: **end for**
-

The Analytical Jacobian Matrix

For practical purposes, it is necessary to find an operator that is able to describe the articular variations of the vector of generalized coordinates \mathbf{q} in the Configuration Space (CS) to the variation of the Cartesian vector \mathbf{X} in the Operational Space (OS) which, as approached in the previous section, can be described by the use of three parameters representing position (x, y, z) and three parameters representing orientation (notation of Euler angles, for instance, ψ, θ, ϕ). The previously introduced geometric Jacobian matrix is able to map the variations of a body-associated frame position ($\dot{p}_c \in \mathfrak{R}^3$) as a function of the variation of the articular configuration ($\dot{q} \in \mathfrak{R}^n$); nevertheless, this geometric Jacobian matrix does not describe the variation of the frame orientation in $SO(3)$, i.e., there is no direct way to relate the angular velocity ω_c obtained in equation (2.29) with the variations in the Euler angles.

This problem is solved by the use of the **Analytical Jacobian Matrix** $J_{A_c}(q) \in \mathfrak{R}^{6 \times n}$, which relates the articular velocities with the linear and angular velocities of the considered frame Σ_c , considering a determined parametric representation of Euler angles. Thus, if the absolute position and orientation of the frame Σ_c is given by:

$$X_c = \begin{pmatrix} p_c \\ \alpha_c \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \\ z_c \\ \psi_c \\ \theta_c \\ \phi_c \end{pmatrix} \quad (2.30)$$

where X_c is called “pose vector”. Then, by differentiating respect to time it is possible to obtain a velocity vector \dot{X}_c , which can be parameterized as a function of the variation in the articular configuration by the use of the Analytical Jacobian matrix as follows:

$$\dot{X}_c = \begin{pmatrix} \dot{p}_c \\ \dot{\alpha}_c \end{pmatrix} = \begin{pmatrix} v_c \\ \dot{\alpha}_c \end{pmatrix} = J_{A_c}(q)\dot{q} \quad (2.31)$$

It is important to note that $\omega_c \neq \dot{\alpha}_c$, therefore the geometric Jacobian matrix cannot be directly used for this purpose. Nonetheless, it is possible to find an operator which relates the geometric Jacobian matrix $J_{G_c}(q)$ with the analytical Jacobian matrix $J_{A_c}(q)$ by:

$$J_{G_c}(q)\dot{q} = \begin{pmatrix} v_c \\ \omega_c \end{pmatrix} = \begin{pmatrix} \dot{p}_c \\ B(\alpha_c)\dot{\alpha}_c \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & B(\alpha_c) \end{pmatrix} \begin{pmatrix} \dot{p}_c \\ \dot{\alpha}_c \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & B(\alpha_c) \end{pmatrix} J_{A_c}(q)\dot{q} \quad (2.32)$$

Then the analytical Jacobian matrix is calculated from the previous equation:

$$J_{A_c}(q) = \begin{pmatrix} I & 0 \\ 0 & [B(\alpha_c)]^{-1} \end{pmatrix} J_{G_c}(q) \quad (2.33)$$

where $I \in \mathfrak{R}^{3 \times 3}$ and $B(\alpha_c) \in \mathfrak{R}^{3 \times 3}$.

The matrix $B(\alpha_c)$ is constituted by the column vectors that represent each rotation, considering the selected Euler angles convention; particularly, according to the z - y - x notation used in this work, the corresponding matrix $B(\alpha_c)$ is given by

$$B(\alpha_c) = \begin{pmatrix} R_z(\psi_c) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & R_z(\psi_c)R_y(\theta_c) \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} & R_z(\psi_c)R_y(\theta_c)R_x(\phi_c) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \quad (2.34)$$

The Mobile Manipulator Jacobian Matrix

The geometric and analytical Jacobian matrices explained so far have a significant limitation; they consider that the kinematic model is provided with a fixed base. This restriction is not suitable for the locomotion of a humanoid robot, since it has a non-inertial (mobile) base. In order to overcome this limitation, there exists a method to obtain the Jacobian matrix for a model with a mobile base.

In order to represent the absolute position of any body that is part of a model with a mobile base, it is important to firstly consider six additional degrees of freedom, described by a pose vector $X_B = q_B$, representing the absolute position and orientation of the model base, as illustrated in Figure 2.27.

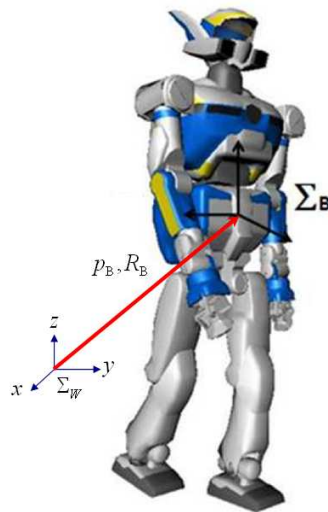


Figure 2.27 - Mobile Base of a kinematic model

In order to consider the base mobility in the geometric model of the robot, these additional degrees of freedom are to be included in the Configuration Space (CS), obtaining a **mobile manipulator vector of generalized coordinates** $q_{mm} \in \mathcal{R}^{6+n}$, as follows:

$$q_{\text{mm}} = \begin{pmatrix} q_B \\ q \end{pmatrix} = \begin{pmatrix} p_B \\ \alpha_B \\ q \end{pmatrix} = \begin{pmatrix} x_B \\ y_B \\ z_B \\ \psi_B \\ \theta_B \\ \phi_B \\ q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix} \quad (2.35)$$

The next step is to determine an operator which relates the variations in this extended vector of generalized coordinates in the Configuration Space (CS) with the linear and angular velocity of any considered body of the kinematic model. This procedure can be divided in a linear part and an angular part.

In the case of the linear part, by the knowledge of the base generalized coordinates q_B , from equations (2.4) and (2.9) it is possible to determine the absolute position vector p_B and the absolute orientation matrix R_B of the model base and thus, the absolute position of any model body j can be represented in terms of the base in the form:

$$p_j = p_B + R_B p_j^B \quad (2.36)$$

where p_j^B represents the relative position of body j , respect to the model base.

The absolute linear velocity of a body j when the model has a mobile base can be deduced by differentiating equation (2.36) respect to time, thus:

$$\dot{p}_j = \dot{p}_B + \dot{R}_B p_j^B + R_B \dot{p}_j^B \quad (2.37)$$

The first term of the previous equation represents the linear velocity of the model base, respect to the world frame. Expressing this term as a function of the variation of the base generalized coordinates in CS:

$$\dot{p}_B = J_{v_B} \dot{p}_B \quad (2.38)$$

It is possible to observe that the value of the linear part of the base geometric Jacobian matrix J_{v_B} is equal to $I^{3 \times 3}$. This statement is true if and only if the geometric model of the base is obtained considering the translation in the three axes, followed by the rotation; i.e., the homogeneous transformation matrix T_B that represent the base translation and rotation must be given, as described in equation (2.4), in the following way:

$$T_B(q_B) = \begin{pmatrix} R_B(\alpha_B) & p_B \\ 0 & 1 \end{pmatrix}$$

In the second term of equation (2.37) the operator \dot{R}_B can be expressed in terms of the base absolute angular velocity vector $\omega_B = (\omega_{x_B} \quad \omega_{y_B} \quad \omega_{z_B})^T$, as follows:

$$\dot{R}_B = \hat{\omega}_B R_B \quad (2.39)$$

where $\hat{\omega}_B \in \mathfrak{R}^{3 \times 3}$ is an anti-symmetric matrix obtained from ω_B , as explained in equation (2.12). The second term of the right side of equation (2.37) can be rewritten to:

$$\begin{aligned} \dot{R}_B p_j^B &= \hat{\omega}_B R_B p_j^B \\ &= -[R_B p_j^B]^\wedge \omega_B \\ &= -[R_B p_j^B]^\wedge J_{\omega_B} \dot{\alpha}_B \end{aligned} \quad (2.40)$$

where $\omega_B = J_{\omega_B} \dot{\alpha}_B$ and, by using the relation between ω_B and $\dot{\alpha}_B$ stated in equation (2.32), it is possible to identify that the angular part of the base geometric Jacobian matrix, which lives in $\mathfrak{R}^{3 \times 3}$, can be expressed as $J_{\omega_B} = B(\alpha_B)$.

The third term of equation (2.37) represents the linear velocity of the body j respect to the model base, expressed in the world frame Σ_W since it is multiplied by the base absolute rotation matrix. By establishing a relationship between the Configuration and Operational spaces, this term can be expressed in the following form:

$$R_B \dot{p}_j^B = R_B J_{v_j}(q) \dot{q} \quad (2.41)$$

where $J_{v_j}(q) \in \mathfrak{R}^{3 \times n}$ is the linear part of the geometric Jacobian matrix of body j in a fixed base condition. The linear velocity in equation (2.37) is then given by:

$$\dot{p}_j = J_{v_B} \dot{p}_B - [R_B p_j^B]^\wedge J_{\omega_B} \dot{\alpha}_B + R_B J_{v_j}(q) \dot{q} \quad (2.42)$$

which can also be written in the following way:

$$\dot{p}_j = \begin{pmatrix} J_{v_B} & -[R_B p_j^B]^\wedge J_{\omega_B} & R_B J_{v_j}(q) \end{pmatrix} \begin{pmatrix} q_B \\ q \end{pmatrix} \quad (2.43)$$

In the case of the angular part, the absolute angular velocity of the body j is given by the sum of the base absolute angular velocity and the relative velocity of body j , respect to the model base, expressed in the world frame. This means that

$$\begin{aligned}
\omega_j &= \omega_B + \omega_j^B \\
&= J_{\omega_B} \dot{\alpha}_B + R_B J_{\omega_j}(q) \\
&= \begin{pmatrix} J_{\omega_B} & R_B J_{\omega_j} \end{pmatrix} \begin{pmatrix} \dot{\alpha}_B \\ q \end{pmatrix}
\end{aligned} \tag{2.44}$$

where $J_{\omega_j}(q) \in \mathfrak{R}^{3 \times n}$ is the angular part of the geometric Jacobian matrix of the body j in a fixed base condition.

Finally, from the previous formulations, it is possible to determine an operator that relates the variations of the new vector of generalized coordinates (which considers the mobile nature of the robot) with the linear and angular velocity of any body of the model. By gathering both equations (2.43) and (2.44) the velocity of body j can be written in the following way:

$$V_j = \begin{pmatrix} v_j \\ \omega_j \end{pmatrix} = \begin{pmatrix} \dot{p}_j \\ \dot{\omega}_j \end{pmatrix} = \text{mm}J_{G_j}(q_{\text{mm}}) \dot{q}_{\text{mm}} \tag{2.45}$$

Where $\text{mm}J_{G_j}(q_{\text{mm}}) \in \mathfrak{R}^{6 \times (6+n)}$ is the **mobile manipulator geometric Jacobian matrix**, associated to body j , and is determined as follows:

$$\text{mm}J_{G_j}(q_{\text{mm}}) = \begin{pmatrix} J_{v_B} & -[R_B p_j^B]^\wedge J_{\omega_B} & R_B J_{v_j}(q) \\ 0 & J_{\omega_B} & R_B J_{\omega_j}(q) \end{pmatrix} \tag{2.46}$$

Furthermore, if an identical formulation as the one established in equation (2.33) is applied to the previous operator, it is possible to construct a matrix that relates the variation in time of the extended vector of generalized coordinates in the Configuration Space (CS) with the variation in time of the position and orientation of any body of the model structure in the Operational Space (OS), considering a mobile base condition. This operator is known as the **mobile manipulator analytical Jacobian matrix** and is given by the following expression:

$$\text{mm}J_{A_j}(q_{\text{mm}}) = \begin{pmatrix} I & 0 \\ 0 & [B(\alpha_j)]^{-1} \end{pmatrix} \text{mm}J_{G_j}(q_{\text{mm}}) \tag{2.47}$$

2.5.6 Inverse Kinematics

In general terms, as illustrated in Figure 2.28, the goal of the inverse geometric modeling is to determine the corresponding value of the vector of generalized coordinates ($q \in \mathcal{R}^n$) in the Configuration Space (CS) that is needed in order to locate a frame Σ_c associated to a body j of the kinematic model at a defined position and orientation in the Operational Space (OS).

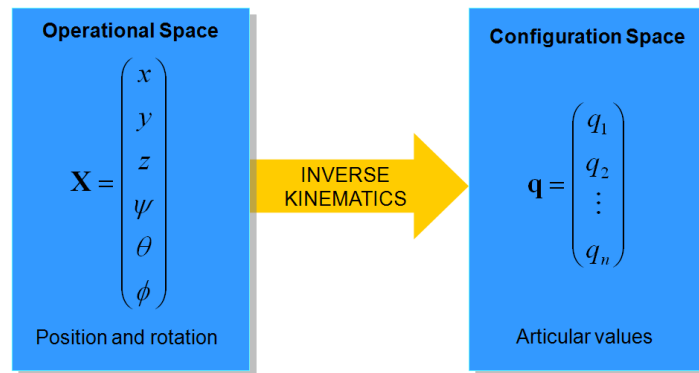


Figure 2.28 Inverse Kinematics maps from the Operational Space (OS) to the Configuration Space (CS)

As previously approached, $X_c \in \mathcal{R}^6$ is a pose vector that can be described as a function of the vector of generalized coordinates as follows:

$$X_c = f(q) \quad (2.48)$$

where X_c is known and q is unknown.

In the particular case where the dimension of the vector q is greater than the dimension of the vector X_c , as it is the case of humanoid robots, then the linear system represented by equation (2.48) becomes a problem with more variables than equations, which means that the system can have an infinite number of solutions, i.e., there are infinite combinations of q that provide the same result value of X_c .

The inverse kinematics can be solved by an analytical or by a numeric approach; generally, the analytic calculation of the inverse geometric model is a much more complex procedure, therefore the numeric approach is more suitable for practical purposes¹. A useful numeric method to perform the inverse kinematics is by using the differential model which was explained in section 2.5.5; in this way, a set of non-linear

¹ Practically every commercial manipulator performs the inverse kinematic procedure by using this method

equations is converted into an equation system with linear expressions, which is easier and faster to solve.

Any task to be performed by the manipulator is represented as a desired point X_{des} in the space (or a set of points forming a trajectory) which is to be reached by a local point $X_c(q)$, associated to a body in the kinematic model. Both points are given as vectors in the Operational Space. The goal is to determine the articular configuration q needed to locate $X_c(q)$ at the position and orientation of X_{des} . This method is performed by the execution of an **iterative algorithm** which seeks to **minimize the error** between the task and the model point, which is given by:

$$e(q) = X_{\text{des}} - X_c(q) \quad (2.49)$$

The numeric inverse kinematics for the considered task is determined from the following actualization rule, according to [9]:

$$q_{k+1} = q_k + \Delta_t \dot{q}_k \quad (2.50)$$

where q_k and q_{k+1} represent the articular configuration for the current and following iteration, respectively; \dot{q}_k represents the articular velocity at the current sample and Δ_t is the time step between subsequent iterations. This numerical approach is illustrated in the control diagram of Figure 2.29

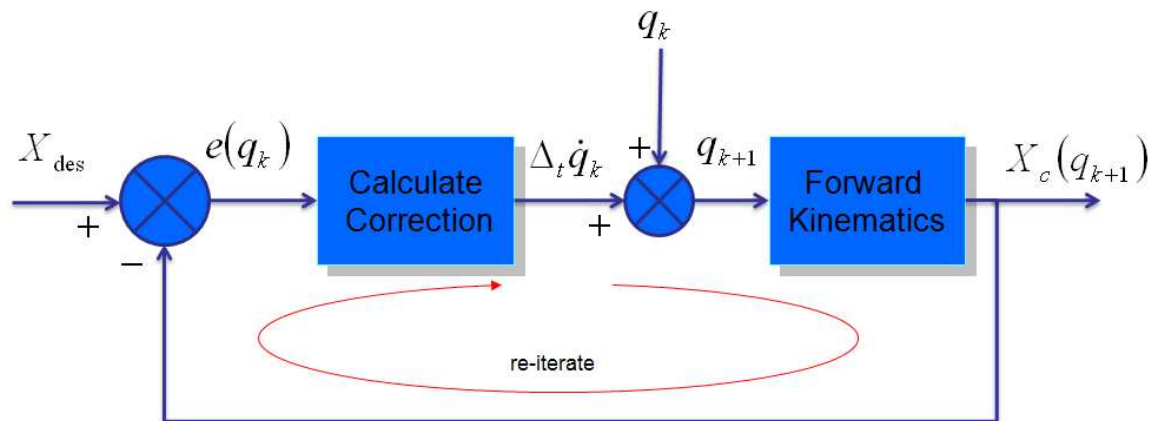


Figure 2.29 - Numerical approach to solve the inverse kinematics

In order to take the model local point $X_c(q)$ to the desired pose X_{des} the previous control scheme must be indefinitely executed; this implies that error will gradually decrease until it becomes small enough, considering a convergence error tolerance Tol_e .

As illustrated in Figure 2.29 four important operations are performed in this iterative algorithm:

- Error measurement
- Calculation of articular correction
- Actualization of articular configuration
- Execution of forward kinematics

The error measurement and the actualization of the articular configuration are performed by the use of equations (2.49) and (2.50); the forward kinematics procedure has been already explained in section 2.5.4; the goal of this section is to present different alternatives to compute the articular correction ($\Delta_t \dot{q}_k$) at each iteration and thus, perform the inverse kinematics of the model.

The articular correction is obtained from two parameters; the first one is the time step Δ_t , which is commonly set as a constant value² and therefore the main objective of this stage of the algorithm is to calculate the value of the articular velocity \dot{q}_k . There are different methods, which are suitable for different purposes and applications.

Numerical Inverse Kinematics (IK)

The first method considers the differential model previously approached in section 2.5.5. As established, the velocity of the frame Σ_c , which is permanently associated to a model body, can be expressed as a function of the articular velocity \dot{q} by the use of a Jacobian matrix (whose type could vary, according to the model nature and parameterization requirements) as follows:

$$v_c = J_c(q)\dot{q} \quad (2.51)$$

It is possible to establish a relationship between the body velocity and the variation in the error in such way that $v_c = \dot{e}(q)$, therefore:

$$\dot{e}(q) = J_c(q)\dot{q} \quad (2.52)$$

Furthermore, it is valid to establish the approximation $\dot{e}(q) \approx e(q)$, then the previous equation can be rewritten as:

$$e(q) \approx J_c(q)\dot{q} \quad (2.53)$$

² There is also the possibility to use a variable time step, according to specific applications; nevertheless in this thesis work the time step among iterations is considered to be always constant.

By solving for \dot{q} it is possible to approximate the value for the articular velocity, as a function of the current iteration error, in the following way:

$$\dot{q}_k \approx [J_c(q_k)]^{-1} e(q_k) \quad (2.54)$$

This procedure is known as **Numerical Inverse Kinematics (IK)**. It is important to note that this method implies that the Jacobian matrix is invertible; i.e. it must be square and non-singular. This is not the particular case of humanoid robots, since they exhibit a redundant nature and, as a result of this, the corresponding Jacobian matrices are not square; nevertheless, the IK is a good tool to approach the methods used for humanoids. A numeric strategy, based in the actualization rule previously stated and the IK method to compute the articular correction is shown in Algorithm 2.3, which considers one kinematic task.

Algorithm 2.3 Numerical Inverse Kinematics (IK) for one task

Input: The task desired pose X_{des} ; maximum allowed amount of iterations Try_{max} ; convergence error tolerance Tol_e and the integration time step Δ_t

Output: Articular value q such that $\|e(q)\| < Tol_e$

- 1: Initialize iterator $k \leftarrow 0$
 - 2: Execute forward kinematics to obtain $X_c(q_k)$
 - 3: Measure error $e(q_k)$
 - 4: **while** ($\|e(q_k)\| > Tol_e$) or ($k < Try_{\text{max}}$) **do**
 - 5: Compute Jacobian matrix $J_c(q_k)$
 - 6: Calculate articular correction $\dot{q}_k \leftarrow [J_c(q_k)]^{-1} e(q_k)$
 - 7: Update articular configuration $q_{k+1} \leftarrow q_k + \Delta_t \dot{q}_k$
 - 8: Execute forward kinematics to obtain $X_c(q_{k+1})$
 - 9: Measure error $e(q_{k+1})$
 - 10: Update $k = k + 1$
 - 11: **end while**
-

Generalized Inverse Kinematics

In the case where the Jacobian matrix is not square (and therefore, not invertible), the calculation of \dot{q} can be formulated as a quadratic problem in the following way:

$$\begin{aligned} \min_{\dot{q}} \quad & f(\dot{q}) = \frac{1}{2} \dot{q}^T W \dot{q} \\ \text{s. t.} \quad & J_j(q) \dot{q} - \dot{e}(q) = 0 \end{aligned} \quad (2.55)$$

where $W \in \mathfrak{R}^{n \times n}$ is a diagonal non-negative matrix. The optimal solution for \dot{q} when $W = I^{n \times n}$ is given by:

$$\dot{q} = J_c(q)^\# \dot{e}(q) \quad (2.56)$$

where $J_c(q)^\#$ is the *pseudo-inverse* matrix of $J_c(q)$ and is obtained by:

$$J_c(q)^\# = J_c^T(q) [J_c(q) J_c^T(q)]^{-1} \quad (2.57)$$

Considering the actualization rule established in equation (2.50) and that $\dot{e}(q) \approx e(q)$ then the equation (2.56) can be written as follows:

$$\dot{q}_k \approx J_c(q_k)^\# e(q_k) \quad (2.58)$$

If in Algorithm 2.3 the operation at step 6 is replaced by the relation stated in the previous equation, the whole procedure acquires the name of **Generalized Inverse Kinematics (GIK)**.

Another important problem of the Inverse Kinematics emerges when **the Jacobian matrix $J_c(q)$ becomes singular**; this situation occurs when one of its singular values is zero due to a specific articular configuration adopted by the model.

When a singularity is present, the linear independency among rows and columns is lost and this condition makes $J_c(q)$ non-invertible and therefore the equation (2.54) cannot be solved. As a result of this, the model body is not able to move and the robot manipulator becomes uncontrollable.

A couple of common examples of singular configurations for a twelve DoF biped robot is shown in Figure 2.30. It is possible to identify from these particular cases that at certain articular configurations, the model end effectors are not able to perform specific movements. For instance, a usual singularity occurs when a leg is in a complete straight configuration (i.e., two of its members share the same center line); this is the reason why the biped robots work with their legs slightly bent.

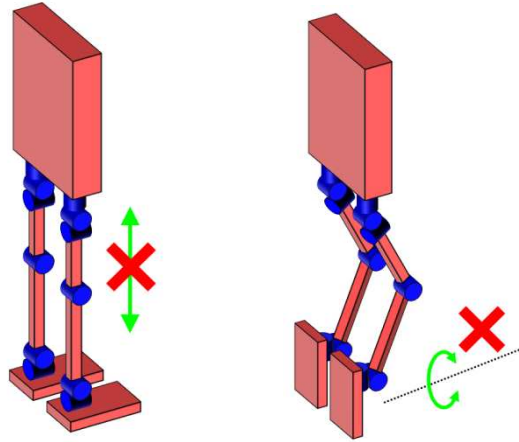


Figure 2.30 - Examples of singular configurations. There exist inaccessible directions for the body end effectors. In these cases, the Jacobian matrix cannot be inverted

The treatment of singularities is still a research topic in Robotics. There are some available solutions to deal with singularities, like that proposed in [10]. This solution consists of introducing a **damping factor** which avoids that any element of the main diagonal of $J_c(q)$ is zero due to the absence of a singular value.

The Jacobian matrix $J_c(q) \in \mathfrak{R}^{n \times m}$ can be represented by its *singular value decomposition* (SVD) as follows:

$$J_c(q) = \sum_{i=1}^r \sigma_i u_i v_i^T \quad (2.59)$$

where r is the range of A ; u_i and v_i are non-zero singular vectors and σ_i is a non-negative scalar singular value.

The pseudo-inverse Jacobian matrix $J_c(q)^\#$, by the use of the SVD is then given by:

$$J_c(q)^\# = \sum_{i=1}^r \frac{1}{\sigma_i} v_i u_i^T \quad (2.60)$$

By the use of the damping factor λ_a , the pseudo-inverse matrix is calculated in the following way:

$$J_c(q)^{\#\lambda_a} = \sum_{i=1}^r \frac{\sigma_i}{\sigma_i^2 + \lambda_a^2} v_i u_i^T \quad (2.61)$$

As formerly established in equation (2.58) this damped pseudo-inverse matrix can be used to determine the optimal solution of \dot{q}_k needed for the actualization rule (2.50); this new optimal value of \dot{q}_k exhibits robustness to singularities and is then given by:

$$\dot{q}_k \approx J_c(q_k)^{\#\lambda_a} e(q_k) \quad (2.62)$$

Prioritized Inverse Kinematics

The inverse kinematics methods approached so far consider the accomplishment of one kinematic task; nevertheless, the redundant nature of the humanoid robots explained in section 2.4.1 implies that only a certain amount of DoF are utilized to meet the task target. In this case, it is possible to take advantage of the resting DoF in the model so they are used to simultaneously accomplish more kinematic tasks.

In order to solve a motion problem, it is necessary that the dimension n of the Configuration Space (CS) is at least equal to the dimension m of the Operational Space (OS). Generally, a kinematic task which requires locating a reference frame Σ_c at a determined position in the 3D space requires three DoF; furthermore, if it is also necessary to locate this frame at a desired orientation, then three additional DoF are required. As a result of this, a minimum amount of six DoF must be available in the CS in order to meet this kind of motion problem. Nonetheless, in the case where $n \gg m$ then the linear system has an infinite number of solutions; this means that the robot is redundant and this redundancy depends on the $n - m$ resting DoF.

Even though the redundancy allows a manipulator to meet two (or more) kinematic tasks at the same time, these tasks might get into conflict and therefore could not be met. This problem can be solved by assigning a level of priority for each kinematic task. This priority scheme is considered for the solution of multi-task problems and is denominated **Prioritized Inverse Kinematics (PIK)**³.

There are basically two different methods utilized to solve the PIK; the first one is proposed in [11] and considers a direct solution to assign an arbitrary number of tasks to a manipulator and the second one, proposed in [12], considers an alternative formulation to also meet an arbitrary number of kinematic tasks, based on the cascade solution of quadratic problems, considering both equality and inequality constraints.

The first scheme [11] proposes a method to assign a secondary kinematic task with a lower priority without affecting the execution of the first task. This is achieved by projecting the action of the second task into the null space of the Jacobian matrix of the first one. From the physical point of view, a variation belonging to the null space of the

³ This scheme also receives the name of Hierarchized Inverse Kinematics (HIK)

Jacobian matrix has no effect on the first task constraint. In order to attain this, the following quadratic problem is to be solved:

$$\begin{aligned} \min_{\dot{q}, \dot{q}_0} \quad & f(\dot{q}, \dot{q}_0) = \frac{1}{2} \|\dot{q} - \dot{q}_0\|^2 \\ \text{s. t.} \quad & J(q)\dot{q} - \dot{e}(q) = 0 \end{aligned} \quad (2.63)$$

where \dot{q} is the articular variation caused by the first task and \dot{q}_0 is the articular variation caused by the second task. The optimal value of \dot{q} that meets the minimization criteria is given by:

$$\dot{q} = J(q)^\# \dot{e}(q) + P(q)\dot{q}_0 \quad (2.64)$$

where $P(q)$ is the projection to the null space of $J(q)$ and is given by:

$$P(q) = I^{n \times n} - J(q)^\# J(q) \quad (2.65)$$

From the previous formulations, by considering a first kinematic task related to the motion of a frame Σ_{c_1} associated to a model body whose Jacobian matrix is $J_1(q)$, the error $e_1(q)$ associated to this task is given by the following expression:

$$\dot{e}_1(q) = J_1(q)\dot{q} \quad (2.66)$$

The optimal value of \dot{q} that minimizes the first task error $e_1(q)$ is represented as follows:

$$\dot{q} = J_1(q)^\# \dot{e}_1(q) + (I - J_1(q)^\# J_1(q))\dot{q}_0 \quad (2.67)$$

If a second task with a lower priority is assigned, then

$$\dot{e}_2(q) = J_2(q)\dot{q} \quad (2.68)$$

By substituting equation (2.67) in the previous equation the following expression is obtained:

$$\dot{e}_2(q) = J_2(q) \left(J_1(q)^\# \dot{e}_1(q) + (I - J_1(q)^\# J_1(q))\dot{q}_0 \right) \quad (2.69)$$

By solving the previous equation for \dot{q}_0 in the following way:

$$\dot{q}_0 = \left(J_2(q) (I - J_1(q)^\# J_1(q)) \right)^\# (\dot{e}_2(q) - J_2(q) J_1(q)^\# \dot{e}_1(q)) \quad (2.70)$$

and substituting the resulting expression of \dot{q}_0 in equation (2.67), the optimal articular variation that meets both kinematic tasks is obtained as follows:

$$\dot{q} = J_1(q)^\# \dot{e}_1(q) + P_1(q) \left((J_2(q) P_1(q))^\# (\dot{e}_2(q) - J_2(q)\dot{q}_1) \right) \quad (2.71)$$

where $P_1(q) = (I^{n \times n} - J_1(q)^\# J_1(q))$ and $\dot{q}_1 = J_1(q)^\# \dot{e}_1(q)$

The previous formulations can be extended to an arbitrary amount p of kinematic tasks by firstly considering a generalization of equation (2.52) in the form:

$$\begin{cases} \dot{e}_1(q) = J_1(q)\dot{q} \\ \dot{e}_2(q) = J_2(q)\dot{q} \\ \vdots \\ \dot{e}_p(q) = J_p(q)\dot{q} \end{cases} \quad (2.72)$$

Then an optimization problem with the following structure is established:

$$\begin{aligned} \min_{\dot{q}_k, w} \quad & f(\dot{q}, w) = \frac{1}{2} \|w\|^2 + \frac{1}{2} \|\dot{q}_p\|^2 \\ \text{s. t.} \quad & J_p(q)P_{p-1}(q)\dot{q}_p - (\dot{e}_p(q) - J_p(q)\dot{q}_{p-1}) = w \end{aligned} \quad (2.73)$$

where w accumulates the operational error of the p kinematic tasks and is minimized (tending to zero) when all tasks are met. The optimal solution for \dot{q} in the previous quadratic problem represents the articular correction for an iteration k , which is needed to compute the inverse kinematics as regarded in Figure 2.29 and Algorithm 2.3, but in this case considering an arbitrary amount of kinematic tasks.

All inverse kinematics schemes proposed so far assume an unconstrained condition of the joint variables; i.e., no **articular limits** have been considered in the methods previously approached. The consideration of the articular limits is indispensable in practical application in order to avoid possible self-collisions.

In [11] a routine known as *clamping* is proposed where, after determining the articular correction \dot{q}_k and updating the articular configuration $q_{k+1} = q_k + \dot{q}_k \Delta_t$, a verification of the articular limits is performed. In the case where the upper or lower articular limit for any joint is violated, the articulation is blocked at that limit and is unblocked until the joint variable is once again inside the limits interval.

The Algorithm 2.4 shows a method to compute the PIK for an arbitrary amount of kinematic tasks considering the basic scheme exposed in Algorithm 2.3, and additionally includes firstly, the optimal solution to the problem (2.73) detailed in lines 5 to 18 and secondly, the articular limits verification and blocking performed by the *clamping* routine in lines 21 to 29.

Algorithm 2.4 Prioritized Inverse Kinematics (PIK) for an arbitrary amount of tasks considering articular limits verification

Input: All tasks desired poses $X_{\text{des}_1}, X_{\text{des}_2}, \dots, X_{\text{des}_p}$; initial configuration q_0 ; maximum allowed amount of iterations Try_{max} ; convergence error tolerance Tol_e and the integration time step Δ_t

Output: Articular value q such that $\|e(q)\| < Tol_e$

- 1: Initialize iterator $k \leftarrow 0$
- 2: Execute forward kinematics to obtain $X_c(q_k)$
- 3: Measure error $\|e(q_k)\| = \sqrt{\|e_1(q)\|^2 + \|e_2(q)\|^2 + \dots + \|e_p(q)\|^2}$
- 4: **while** ($\|e(q_k)\| > Tol_e$) or ($k < Try_{\text{max}}$) **do**
- 5: Initialize $P_1(q) \leftarrow I^{n \times n}$
- 6: Initialize $\dot{q}, \dot{q}_1 \leftarrow 0$
- 7: Initialize $q_k \leftarrow \text{Free}$
- 8: Initialize $LimitViolation \leftarrow \text{False}$
- 9: **while** $LimitViolation \leftarrow \text{False}$ **do**
- 10: **for** $i = 1, 2, \dots, p$ **do**
- 11: Compute Jacobian matrix $J_i(q)$
- 12: Measure task error $e_i(q)$
- 13: Calculate $\hat{J}_i(q) \leftarrow J_i(q)P_i(q)$
- 14: Calculate $\hat{J}_i(q)^\#$
- 15: $\dot{q}_{i+1} \leftarrow \dot{q}_i + \hat{J}_i(q)^\#(e_i(q) - J_i(q)\dot{q}_i)$
- 16: Update $P_{i+1} \leftarrow P_i - \hat{J}_i(q)^\# \hat{J}_i(q)$
- 17: **end for**
- 18: Calculate articular correction $\dot{q}_k \leftarrow \dot{q}_{p+1}$
- 19: Update articular configuration $q_{k+1} \leftarrow q_k + \Delta_t \dot{q}_k$
- 20: $LimitViolation \leftarrow \text{False}$
- 21: **for all** q_{j_k} Free joint in $j = 1, 2, \dots, n$ **do**
- 22: **if** (update of \dot{q}_{j_k} violates limit \dot{q}_{L_j}) **then**
- 23: $LimitViolation \leftarrow \text{True}$
- 24: $\dot{q}_{j_k} \leftarrow \dot{q}_{L_j}$

```

25:           Diagonal term  $j$  of  $P_1(q) \leftarrow 0$ 
26:            $q_{j_k} \leftarrow$  Blocked
27:           end if
28:       end for
29:       Execute forward kinematics to obtain  $X_c(q_{k+1})$ 
30:       Measure error  $e(q_{k+1})$ 
31:       Update  $k \leftarrow k + 1$ 
32:   end while
33: end while

```

The method proposed above has the main advantage of being only able to deal with equality constraints, as established in system (2.72). Nevertheless, for practical purposes it is often necessary to attain inequality tasks as well. The second scheme to compute the model PIK, proposed in [12], considers the computation of the inverse kinematics based in the cascade solution of a series of quadratic problems, considering both systems of linear equalities and inequalities.

Let A and C be matrices in $\mathfrak{R}^{m \times n}$; q a vector in \mathfrak{R}^n ; b and d vectors \mathfrak{R}^m . A system of linear equalities can be expressed in the following way:

$$Ax = b \quad (2.74)$$

and a system of linear inequalities is expressed as:

$$Cx \leq d \quad (2.75)$$

A system of linear equalities may have no solution or may define an affine subspace of \mathfrak{R}^n (e.g., in case $n = 3$, this affine subspace could be a point, a line, a plane or the whole space \mathfrak{R}^3). A linear inequality, when it has solutions, defines a half space. The set of solutions of a system of linear inequalities is the intersection of the halfspaces generated by its inequalities.

A kinematic task can be represented, according to its nature, as a system of linear equalities or inequalities, by considering that in systems (2.74) and (2.75):

- x is the Articular configuration q ,
- A is the Jacobian matrix of a equality task $J_E(q)$,
- b is the error time derivative of a equality task $\dot{e}_E(q)$,
- C is the Jacobian matrix of a inequality task $J_I(q)$ and
- d is the error time derivative of an inequality task $\dot{e}_I(q)$.

The mathematical representation of kinematic tasks as systems of linear equalities or inequalities is detailed, for a series of practical applications, in section 1. .

When trying to satisfy a system (2.74) of linear equalities, the set of optimal solutions is given by the following minimization problem:

$$\begin{aligned} \min_{x \in \mathfrak{R}^n, w \in \mathfrak{R}^m} \quad & \frac{1}{2} \|w\|^2 \\ \text{s. t.} \quad & w = Ax - b \end{aligned} \quad (2.76)$$

In the case of a system (2.75) of linear inequalities, the set of optimal solutions is given by the minimization problem:

$$\begin{aligned} \min_{x \in \mathfrak{R}^n, w \in \mathfrak{R}^m} \quad & \frac{1}{2} \|w\|^2 \\ \text{s. t.} \quad & w \geq Cx - d \end{aligned} \quad (2.77)$$

Both optimization problems (2.76) and (2.77) have similar layouts and properties. The generalization of these results to mixed systems of linear equalities and inequalities is considered in another minimization problem, which in a more compact form is expressed as:

$$\begin{aligned} \min_{x \in \mathfrak{R}^n, w \in \mathfrak{R}^m} \quad & \frac{1}{2} \|Ax - b\|^2 + \frac{1}{2} \|w\|^2 \\ \text{s. t.} \quad & Cx - w \leq d \end{aligned} \quad (2.78)$$

Now, when attempting to satisfy a set of kinematic tasks with a strict level of priority among these tasks, at each level of priority both a system of linear equalities (2.74) and (2.75) are considered, with matrices and vectors A_i , b_i , C_i , d_i indexed by their priority level i .

In [12], they propose to do so by solving at each level of priority a minimization problem such as (2.78), with levels of priority decreasing with i . At each priority level, these systems are satisfied while strictly enforcing the solutions found for the levels of higher priority. The hierarchy is ensured by minimizing the error of task i (i.e., $e_i(q) = X_{\text{des}_i} - X_i(q) \rightarrow 0$) in the null-space of the Jacobian matrix associated to task $i-1$. Using this hierarchical set of equalities and inequalities, the mixed quadratic problem is formulated as follows:

$$\begin{aligned}
S_i &= \begin{cases} \bar{A}_i x = \bar{b}_i \\ \bar{C}_i x \leq \bar{d}_i \end{cases} \\
S_{i+1} &= \begin{cases} \arg \min_{x \in \mathbb{R}^n, w \in \mathbb{R}^m} \frac{1}{2} \|A_i x - b_i\|^2 + \frac{1}{2} \|w\|^2 + \frac{1}{2} \|\lambda x\|^2 \\ \text{s. t. } C_i x - w \leq d_i \end{cases} \quad (2.79)
\end{aligned}$$

where $S_i \subseteq S_{i+1}$ represents the admissible solution set for task $i+1$, according to its priority and S_i is an *argumented* linear system that maintains the solution of upper tasks. Thus, after solving the mixed quadratic problem (2.79) for the priority level i and obtaining the optimal solution x_i^* , S_{i+1} becomes

$$\begin{aligned}
\bar{A}_{i+1} &= \begin{bmatrix} \bar{A}_i \\ A_i \end{bmatrix} \\
\bar{b}_{i+1} &= \begin{bmatrix} \bar{b}_i \\ A_i x_i^* \end{bmatrix} \quad (2.80)
\end{aligned}$$

which forms a non-empty convex polytope. Furthermore, all inequalities are evaluated after each call; if a subset of them is violated, then all inequalities of this subset are placed in $\{\bar{A}_{i+1}, \bar{b}_{i+1}\}$ or placed in $\{\bar{C}_{i+1}, \bar{d}_{i+1}\}$ otherwise. The Algorithm 2.5 processes the priority levels from highest to lowest and solves at every level the corresponding quadratic problem.

Algorithm 2.5 Solution of cascade quadratic problems, incorporating mixed systems of linear equalities and inequalities

Input: Total amount of tasks p ; equality and inequality constraints $\{A_i, b_i, C_i, d_i\}$ for all levels of priority

Output: Optimal value for x to satisfy all systems while ensuring priority

- 1: Initialize the system of linear equalities $\{\bar{A}_0, \bar{b}_0\}$ to empty.
- 2: Initialize the system of linear equalities $\{\bar{C}_0, \bar{d}_0\}$ to empty.
- 3: **for** $i=0$ to $p-1$ **do**
- 4: Solve the quadratic problem (2.79) to obtain x_i^*
- 5: $\bar{A}_{i+1} \leftarrow \begin{bmatrix} \bar{A}_i \\ A_i \end{bmatrix}$, $\bar{b}_{i+1} \leftarrow \begin{bmatrix} \bar{b}_i \\ A_i x_i^* \end{bmatrix}$
- 6: $\bar{C}_{i+1} \leftarrow \bar{C}_i$, $\bar{d}_{i+1} \leftarrow \bar{d}_i$

```

7:      for all  $c_i^j$  in  $C_i$  do
8:          if  $c_i^j x_i^* \leq d_i^j$  then
9:               $\bar{C}_{i+1} \leftarrow \begin{bmatrix} \bar{C}_{i+1} \\ c_i^j \end{bmatrix}$ ,  $\bar{d}_{i+1} \leftarrow \begin{bmatrix} \bar{d}_{i+1} \\ d_i^j \end{bmatrix}$ 
10:          else
11:               $\bar{A}_{i+1} \leftarrow \begin{bmatrix} \bar{A}_{i+1} \\ c_i^j \end{bmatrix}$ ,  $\bar{b}_{i+1} \leftarrow \begin{bmatrix} \bar{b}_{i+1} \\ c_i^j x_i^* \end{bmatrix}$ 
12:          end if
13:      end for
14:  end for

```

In the practice, the previous algorithm is included as the priority loop of a more general PIK scheme, in order to obtain the optimal solution of the articular correction \dot{q}_k as originally regarded in Figure 2.29. Particularly, it is possible to replace lines 5 to 18 of Algorithm 2.4 with the scheme proposed in Algorithm 2.5 and exclude the clamping routine, since the articular limits constraint can be represented as one type of inequality task, as it will be explained in section 1. .

It is important to note that the method proposed above incorporates explicit calls for the solution of the problem (2.79), which can be obtained by using any kind of informatics package (e.g., MATLAB Quadratic Programming Toolbox [13], qpOASES [14], etc.) dedicated to the solution of quadratic programs, or there exist also works like [15] and [16].

2.6 Dynamics of humanoid robots

The previous subchapter has dealt with the problem of robot kinematics, presenting a method to calculate the position and orientation of a body of the model structure as a function of the articular configuration (forward Kinematics), and also approaching a series of techniques to do the reciprocal process (inverse Kinematics). Nonetheless, when the robot is in motion, it is necessary to take into account other phenomena concerning the robot dynamic behavior. These phenomena can be neglected whether the movements that are performed by the manipulator are slow⁴ enough; nevertheless when treating with biped locomotion it is impossible to ignore the system dynamics. This subchapter analyses the dynamics of a multi-body system and explains the concept of a physic quantity of high significance in humanoid Robotics: the Zero-Moment Point (ZMP).

2.6.1 Basic Concepts in robot Dynamics

Physical Parameters

The Table 2-1 summarizes the four physical parameters that play an important role in robot Dynamics. Through this section, the concepts of these terms and their influence in the system behavior are explained.

A humanoid robot is constituted by a complex structure which can be regarded as an assembly of a finite number of bodies. Being N the total amount of bodies that compose the robot and m_j the mass of body j , the **total mass** of the robot is given by:

$$M = \sum_{j=1}^N m_j \quad (2.81)$$

Moreover, considering c_j as the absolute position of the CoM of a body j regarded as a punctual mass, the **position of the total CoM** of the robot can be expressed as follows:

$$c = \frac{1}{M} \sum_{j=1}^N m_j c_j \quad (2.82)$$

⁴ The term *slow*, although being evidently subjective, is used in the practice to describe movements that are conceived as a succession of fixed configurations, i.e., the displacements are *quasi-static*.

Table 2-1 Important physical parameters in robot Dynamics			
Physical Parameter	Basic definition	Representation	Unit
Mass	The sum of the masses of all bodies that constitute the kinematic model	M	Kg
Position of the Center of Mass (CoM)	The position of the center of gravity of the whole model	$c = \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix}$	m
Linear Momentum	Measurement of the robot translation movements	$\mathcal{P} = \begin{pmatrix} \mathcal{P}_x \\ \mathcal{P}_y \\ \mathcal{P}_z \end{pmatrix}$	N · s
Angular Momentum	Measurement of the robot rotation movements, around the origin	$\mathcal{L} = \begin{pmatrix} \mathcal{L}_x \\ \mathcal{L}_y \\ \mathcal{L}_z \end{pmatrix}$	N · m · s

From equation (2.15) it is possible to determine the value of c_j when previously knowing in one hand, the absolute position p_j and orientation R_j of a local frame Σ_j associated to body j and in the other hand, the relative position \bar{c}_j of the body j CoM, respect to this local frame. In a similar way, it is possible to determine the absolute velocity of the center of mass of the body j , which is noted by \dot{c}_j , but in this case by the knowledge of R_j , \bar{c}_j and from equations (2.16) and (2.17) the absolute linear and angular velocities v_j and ω_j of the body local frame Σ_j , respectively; thus:

$$\dot{c}_j = v_j + \omega_j \times (R_j \bar{c}_j) \quad (2.83)$$

The total **linear momentum** of the robot can be expressed as the sum of all bodies' linear momentum, where each of them is given by the product of the body mass and the velocity of its CoM, as follows:

$$\mathcal{P} = \sum_{j=1}^N m_j \dot{c}_j \quad (2.84)$$

In the same way, the total **angular momentum** of the robot is given by the sum of the angular momentum generated by the movement of all bodies constituting the kinematic model, expressed at the origin of the reference frame:

$$\mathcal{L} = \sum_{j=1}^N \mathcal{L}_j \quad (2.85)$$

where \mathcal{L}_j is the angular momentum generated by the j -th body and is given by:

$$\mathcal{L}_j = c_j \times \mathcal{P}_j + R_j \bar{I}_j R_j^T \omega_j \quad (2.86)$$

The parameter \bar{I}_j represents the inertial tensor of the body, expressed in the local coordinates frame. Since all parameters should be expressed in the reference frame, the local inertial tensor must be expressed as an absolute parameter; therefore the expression $R_j \bar{I}_j R_j^T$ is used in equation (2.86) to describe the absolute inertial tensor

There is a common misconception about the angular momentum, which consists of considering that it is defined only by a rotation movement. By regarding the previous equation, it is possible to identify that the angular momentum is composed by two terms; the first term is related to the effect generated by the body linear displacement and the second one incorporates the effect of the body angular displacement. Thus, when a body describes a linear motion, even in absence of an angular velocity ω_j it can exhibit an angular momentum described by $c_j \times \mathcal{P}_j$, due to the law of conservation of the angular momentum. Figure 2.31 illustrates, in a general way, the statement just described by showing a relationship between the linear and angular momentums of a rigid body in motion, represented by a punctual mass.

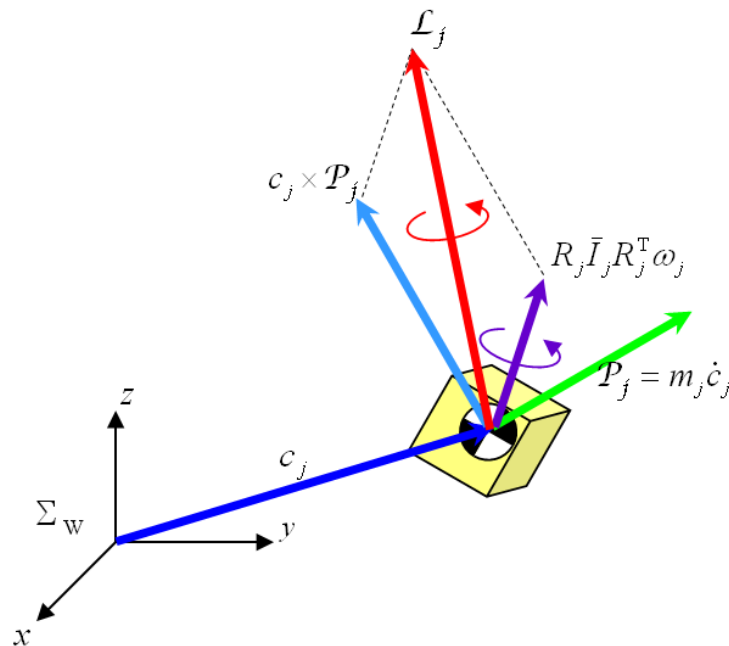


Figure 2.31 Relationship between linear and angular momentums of a rigid body in motion

System dynamics

The **system dynamics** allows establishing a relationship among the different physical parameters defined in Table 2-1; this relationship is grouped in the three following equations:

$$\dot{c} = \frac{\mathcal{P}}{M} \quad (2.87)$$

$$\dot{\mathcal{P}} = f_{all} \quad (2.88)$$

$$\dot{\mathcal{L}} = \tau_{all} \quad (2.89)$$

The equation (2.87) relates the velocity of the CoM and the linear momentum, in the case of a translation movement. This equation establishes that the total linear momentum can be given by the product of the model total mass by the CoM velocity. Besides, the equation (2.88) indicates that in the case of a translation movement, the linear momentum depends on the resultant **external forces** that are applied to the robot. It can be noted that by differentiating equation (2.87) respect to time, solving for $\dot{\mathcal{P}}$ and substituting the resulting expression in equation (2.88), the second Newton's law $f_{all} = M \cdot \ddot{c}$ is obtained.

There exist several kinds of external forces that can be exerted on the robot structure. One of these forces is the **gravitational force** that is applied to the CoM of the whole robot, which represents the sum of the gravitational forces associated to all the bodies in the model. This gravitational force is always present and does not depend on the robot motion, therefore it is generally treated in a separated way from other external forces, and this implies that:

$$f_{all} = MG + f = M \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + f \quad (2.90)$$

where $G = (0 \ 0 \ -g)^T$ is a gravitational acceleration vector whose components describe the attraction that maintains the bodies on the surface of a planet⁵ and f groups all the resting external forces that are applied to the robot.

It is important to note that the gravitational acceleration vector G presented in equation (2.90) implies that the gravitational force acts only on the direction of $-z$. From a certain perspective, this statement is false since the shape of the planets is round and

⁵ The gravitational acceleration vector G incorporates the constant acceleration of gravity g which can exhibit different values, according to the location in the Universe; it is known, for instance, that g is equal to 9.81 m/s^2 on planet Earth, 1.7 m/s^2 on the Moon, or 3.6 m/s^2 on Mars.

furthermore, multiple gravitational forces are exerted among all existing bodies in the Universe, each of these depending on the bodies' masses and the distance among them. However, these arguments can be discarded in the view of the fact that, when considering that the robot is located on a planet surface (for instance, the Earth), the gravitational forces exerted on the robot by other bodies in the Universe are evidently negligible in comparison with the gravitational force that attracts the robot to the planet. Moreover, by taking into account the relatively large size of the planet respect to the robot, its surface can be regarded to be flat from a local point of view. As a result of these considerations, it is valid to assume that the gravitational effects on the robot structure are exerted only in the direction of $-z$. Obviously, this assumption is not valid any more if the robot is situated in a location which is not a planet surface; nevertheless, this practical case is not the objective of the present research and therefore is not taken into account for this thesis work.

By taking together equations (2.88) and (2.90), the linear momentum can be expressed as follows:

$$\dot{\mathcal{P}} = MG + f \quad (2.91)$$

As mentioned above the vector f groups all the external forces, beside gravity, that are applied to the robot structure; i.e., the contact forces with the environment. In this particular case of study, f is focused on the **reaction forces exerted by the ground** on the robot support points⁶.

When a robot maintains an immobile position, the variation of the linear momentum is null and the gravity force are equilibrated with the ground reaction forces. If these reaction forces disappeared, the linear momentum would increase rapidly due to the action of the gravitational force; in other words, the robot would be in a *free fall* condition.

By its part, the equation (2.89) indicates that the angular momentum depends on the **resultant moments of the external forces** τ_{all} . As occurs in the case of equation (2.90), a part of the external moments exists due to gravitational effects. The moment that is present in the robot due to the gravitational force is called **gravitational moment** and is given by the relation:

$$\tau_g = c \times MG \quad (2.92)$$

Being τ the moments of external forces excluding gravity, the resulting moment of external forces applied to the robot is given by:

⁶ Other examples of external forces that can be applied on the robot are the reaction force when pushing an object or the force exerted by the wind; nonetheless, these cases are not approached in this work.

$$\tau_{all} = \tau_g + \tau = c \times MG + \tau \quad (2.93)$$

In this way, the equation describing the rotation movement around the origin can be written as:

$$\dot{L} = c \times MG + \tau \quad (2.94)$$

As stated above regarding the external forces, in this particular case of study only the **reaction moments of the ground** are considered in τ . When a robot is immobile, this moment is equilibrated with the gravitational moment; in the opposite case, the reaction moment of the ground increases quickly and the robot tends to *fall*.

2.6.2 The Zero-Moment Point (ZMP)

The base of most industrial manipulators is fixed to the ground; therefore their operational space is reduced only to the articular capabilities; nonetheless, in the case of humanoid robots, whose base is mobile, the only contact with the ground is at the surface of the feet soles. Due to this lack of fixed condition, any type of humanoid locomotion must be performed while ensuring stability; i.e., maintaining the contact with the ground. The use of the ZMP is a commonly utilize method to achieve this goal. This section introduces in a general way the concept of ZMP, according to [17] and the next section presents a method to approximate its position, by using the formulations obtained in the previous section.

Definition of ZMP

Consider the case shown in Figure 2.32, showing an example of a distributed force f beneath the sole of a humanoid foot. This force describes the contact between the robot foot and the ground. Since the sign of the distributed force is the same through the whole surface, it is possible to represent the force f as a resulting force f_R , where the application point is situated on the foot surface limit. This point defines the Zero-Moment Point (ZMP).

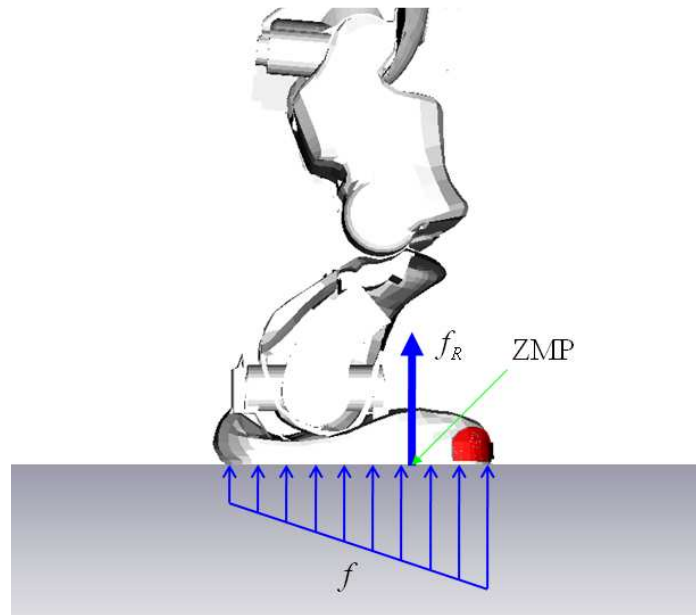


Figure 2.32 – Definition of the Zero-Moment Point (ZMP)

ZMP and the Support Polygon

An important concept in the definition of the ZMP is the support polygon⁷. From the mathematic point of view, the support polygon of a body is defined as the convex hull that surrounds the set of contact points between the ground and the body surface⁸. Figure 2.33 illustrates three different cases of contact between the robot feet and the ground, remarking the support polygon surface and perimeter. From a more intuitive perspective, by placing an elastic band around the robot feet at the level of the contact surfaces, the shape obtained represents the support polygon perimeter.

Furthermore, it is possible to establish a relationship between these two presented concepts and it is based in the fact that **the ZMP is always located inside the support polygon**⁹. This important statement is the start point for all mathematical foundations.

⁷ The term support polygon is also known as sustentation polygon. Both denominations can be used indistinctly.

⁸ A further explanation about the concept of convex hull is detailed in Appendix C.

⁹ According to [34] the ZMP can exist outside the sustentation polygon; nevertheless, the majority in the Robotics research community affirms that the ZMO can never exist outside this region. In [17] an explanation defending the existence of the ZMP only inside the support polygon is presented. The discussions about this topic are still frequent.

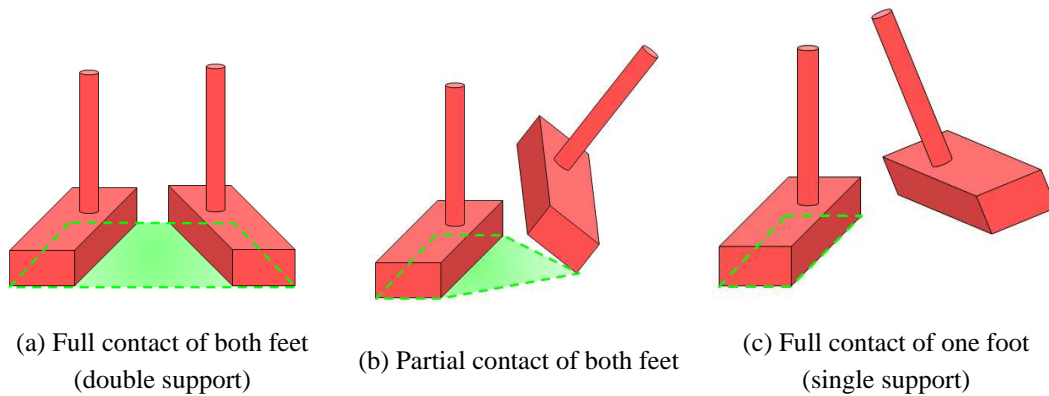


Figure 2.33 – Representation of the support polygon

Additionally, there is also an important parameter that is needed to establish a relationship among the center of mass (CoM) of a body, the ZMP and the support polygon. This parameter is called **CoM projection to the ground** and it is considered as the intersection point between the CoM gravity line and the ground surface.

Consider the case of Figure 2.34(a) where the CoM projection is located inside the support polygon; in this situation, the robot keeps its stability by only maintaining this condition, which is commonly the case of quasi-static locomotion control. In the particular case where the robot remains immobile, the position of the CoM projection coincides with the position of the ZMP; in contrast, when the robot performs dynamic movements, the CoM projection does not coincide with the ZMP and it can also be located outside the support polygon, which is the case of Figure 2.34(b).

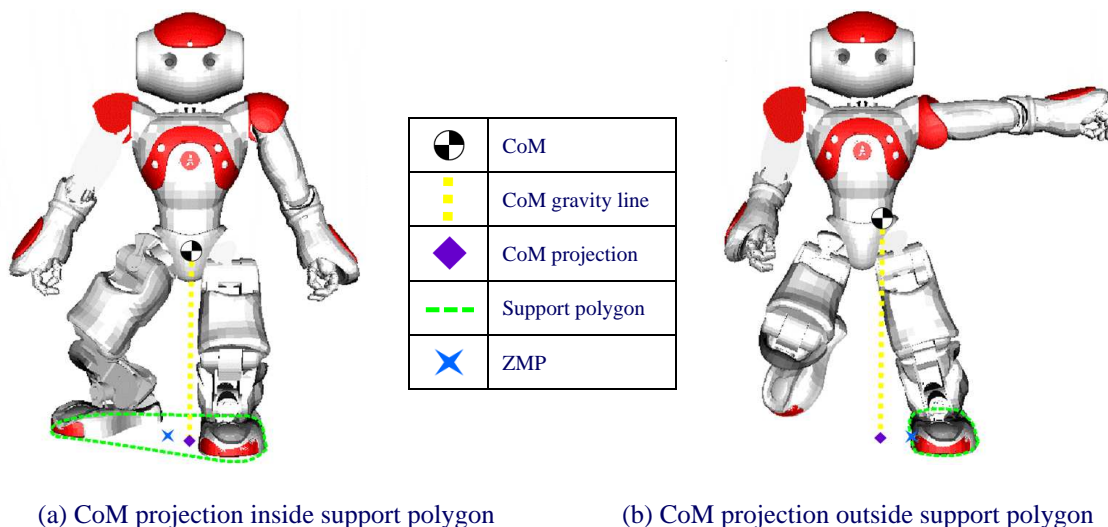


Figure 2.34 – Relationship among CoM, ZMP and support polygon

2.6.3 Determination of the position of the ZMP

Ground reaction forces on the robot surface

The ground exerts a distributed reaction force on the robot soles, which can be represented as vectors in the 3D space applied at infinitesimal elements of the contact surface. As illustrated in Figure 2.35, Being $r = (\xi \ \eta \ 0)^T$ the vector that describes the absolute position of any point on the contact surface S , the reaction force that is applied to this point (represented as an infinitesimal element dS) can be decomposed into a vertical force $\rho(\xi, \eta)$ and two horizontal forces $\sigma_x(\xi, \eta)$ and $\sigma_y(\xi, \eta)$ for simplification purposes; in a real situation, all components are applied simultaneously.

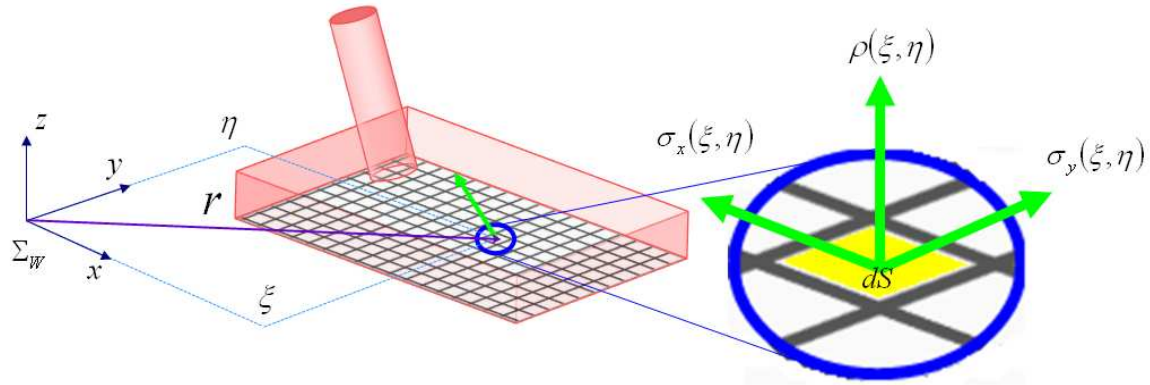


Figure 2.35 – Tridimensional decomposition of the reaction force applied on a surface differential dS .

By taking together the vertical forces $\rho(\xi, \eta)$ that are applied on all the surface infinitesimal elements, it is possible to determine the resulting force f_z applied to any given point $p = (p_x \ p_y \ 0)^T$ on the contact surface. This resulting force is equivalent to the whole force distribution and is determined as follows:

$$f_z = \int_S \rho(\xi, \eta) dS \quad (2.95)$$

where the operator \int_S defines the integration on the contact surface S between the robot foot sole and the ground. Moreover, it is possible to calculate the resulting moment $\tau_n(p)$ of the vertical components of all reaction forces on the surface at the given point p , and it is defined in the following way:

$$\tau_n(p) = \begin{pmatrix} \tau_{nx} \\ \tau_{ny} \\ \tau_{nz} \end{pmatrix} \quad (2.96)$$

$$\tau_{n_x} = \int_S (\eta - p_y) \rho(\xi, \eta) dS \quad (2.97)$$

$$\tau_{n_y} = - \int_S (\xi - p_x) \rho(\xi, \eta) dS \quad (2.98)$$

$$\tau_{n_z} = 0 \quad (2.99)$$

By using the previous formulations, it is possible to determine the coordinates p_x and p_y of the point p , where resulting moment $\tau_n(p)$ becomes zero. By using equations (2.97) and (2.98), these values can be calculated by the following relations:

$$p_x = \frac{\int_S \xi \rho(\xi, \eta) dS}{\int_S \rho(\xi, \eta) dS} \quad (2.100)$$

$$p_y = \frac{\int_S \eta \rho(\xi, \eta) dS}{\int_S \rho(\xi, \eta) dS} \quad (2.101)$$

This point where the resulting moment is annulled defines the *center of pressure* of the contact surface, or in other terms, the **Zero-Moment Point**.

Besides, by considering the horizontal components $\sigma_x(\xi, \eta)$ and $\sigma_y(\xi, \eta)$ of the ground reaction force acting on all infinitesimal surface elements, it is also possible to determine the resulting forces f_x and f_y that are applied to any given point p on the contact surface, respectively:

$$f_x = \int_S \sigma_x(\xi, \eta) dS \quad (2.102)$$

$$f_y = \int_S \sigma_y(\xi, \eta) dS \quad (2.103)$$

The resulting moment of the horizontal reaction forces on point p is given by:

$$\tau_t(p) = \begin{pmatrix} \tau_{t_x} \\ \tau_{t_y} \\ \tau_{t_z} \end{pmatrix} \quad (2.104)$$

$$\tau_{t_x} = 0 \quad (2.105)$$

$$\tau_{t_y} = 0 \quad (2.106)$$

$$\tau_{t_z} = \int_S ((\xi - p_x) \sigma_y(\xi, \eta) - (\eta - p_y) \sigma_x(\xi, \eta)) ds \quad (2.107)$$

The analysis of this section determines that, in one hand, the vertical components of the reaction forces generate the horizontal components of the resulting moment and in the other hand, the horizontal components of the reaction forces produce the vertical component of

the resulting moment. Thus, from the previous formulations, the ground reaction force distribution on the robot foot sole can be replaced by an equivalent resulting force

$$f = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}, \quad (2.108)$$

and a resulting moment

$$\tau_p = \tau_n(p) + \tau_i(p) = \begin{pmatrix} 0 \\ 0 \\ \tau_{iz} \end{pmatrix}, \quad (2.109)$$

both applied at the Zero-Moment Point p .

By regarding the previous equations, it is possible to identify that when the robot is in motion in the tridimensional space, the vertical component of the resulting moment is generally a non-zero value; therefore, the most accurate definition of the ZMP corresponds to **the point where the horizontal components of the resulting moment of the ground reaction force are annulled.**

Calculation of the ZMP by using force sensors

The most common method to measure the position of the ZMP is by using a set of force sensors that are fixed at the foot soles of a real humanoid robot. By using these sensors it is possible to have a finite set of reaction forces, which can represent the distributed reaction force between the ground and the robot soles, as explained at the beginning of this section. The more sensors are placed at the robot feet, the more accurate the representation of the distributed force will be.

In order to measure the position of the ZMP of a biped walking robot, there are two different situations that must be considered:

- Measure the ZMP considering the reaction force between one foot and the ground (case of single support).
- Measure the ZMP considering the reaction force between both feet and the ground (case of double support).

Consider the model of Figure 2.36, schematizing a humanoid foot in contact with the ground. The forces and moments applied by the ground to the foot sole are measured by a set of force sensors. While the robot is in motion and the feet contact with the ground is ensured, the position of the ZMP can be calculated from the output signal of these sensors.

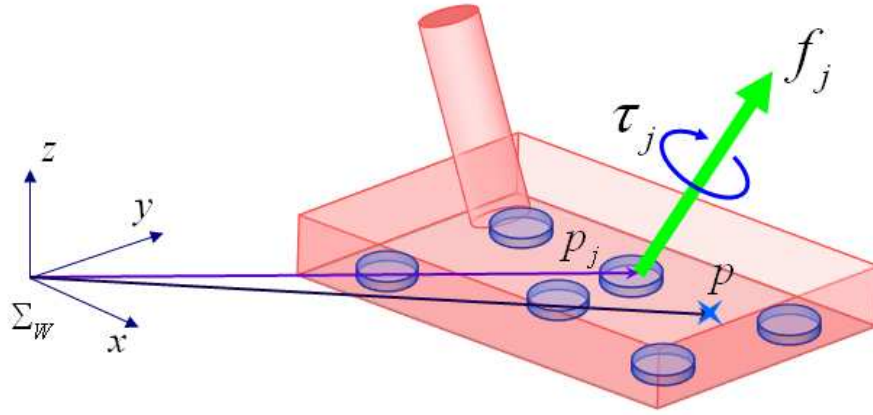


Figure 2.36 – Definition of the position of the ZMP as a function of the position and output reading of a set of force sensors

At points p_j (where $j = \{1, 2, \dots, N\}$), the forces f_j and the moments τ_j are measured, respect to the world frame Σ_W . By taking all this input information together, the resulting force and moment applied at a given point $p = (p_x \ p_y \ p_z)^T$ are given by:

$$f(p) = \sum_{j=1}^N f_j \quad (2.110)$$

$$\tau(p) = \sum_{j=1}^N (p_j - p) \times f_j + \tau_j \quad (2.111)$$

The position of the ZMP can be approximated by fixing to zero the x and y components of the moment $\tau(p)$ from equation (2.111) and then solving for p_x and p_y , as follows:

$$p_x = \frac{\sum_{j=1}^N (-\tau_{j_y} - (p_{j_z} - p_z) f_{j_x} + p_{j_x} f_{j_z})}{\sum_{j=1}^N f_{j_z}} \quad (2.112)$$

$$p_y = \frac{\sum_{j=1}^N (\tau_{j_x} - (p_{j_z} - p_z) f_{j_y} + p_{j_y} f_{j_z})}{\sum_{j=1}^N f_{j_z}} \quad (2.113)$$

where

$$f_j = \begin{pmatrix} f_{j_x} \\ f_{j_y} \\ f_{j_z} \end{pmatrix}, \tau_j = \begin{pmatrix} \tau_{j_x} \\ \tau_{j_y} \\ \tau_{j_z} \end{pmatrix} \text{ and } p_j = \begin{pmatrix} p_{j_x} \\ p_{j_y} \\ p_{j_z} \end{pmatrix}$$

The previous calculations consider that the humanoid is making contact with the ground by using only one foot, also known as single support. In the case of double support, an identical technique as the one proposed in equations (2.112) and (2.113) is firstly performed, obtaining a resulting reaction force (f_R, f_L) and a ZMP position (p_R, p_L) for each foot from the input readings of the force sensors. Secondly, the global position of the ZMP is approximated by taking together the obtained information for each foot, as follows:

$$p_x = \frac{p_{R_x} f_{R_z} + p_{L_x} f_{L_z}}{f_{R_z} + f_{L_z}} \quad (2.114)$$

$$p_y = \frac{p_{R_y} f_{R_z} + p_{L_y} f_{L_z}}{f_{R_z} + f_{L_z}} \quad (2.115)$$

where

$$f_R = \begin{pmatrix} f_{R_x} \\ f_{R_y} \\ f_{R_z} \end{pmatrix}, f_L = \begin{pmatrix} f_{L_x} \\ f_{L_y} \\ f_{L_z} \end{pmatrix}, p_R = \begin{pmatrix} p_{R_x} \\ p_{R_y} \\ p_{R_z} \end{pmatrix} \text{ and } p_L = \begin{pmatrix} p_{L_x} \\ p_{L_y} \\ p_{L_z} \end{pmatrix}$$

Calculation of the ZMP based on the robot movements

By using all the theories and formulations presented at the beginning of this section and the system dynamics equations approached in section 2.6.1, it is also possible to approximate the position of the ZMP as a function of the given robot movements.

As approached in equations (2.108) and (2.109), the reaction forces between the ground and the robot feet surface can be represented as a resulting equivalent force f and a resulting moment τ_p applied at the Zero-Moment Point p ; moreover, from the formulation presented in equation (2.111) it is possible to calculate the resulting moment τ , expressed at the origin of the world reference frame Σ_w in the following way:

$$\tau = p \times f + \tau_p \quad (2.116)$$

Besides, it is possible to utilize the previous equation and the dynamic relations shown in equations (2.91) and (2.94), and manipulate them in order to establish a new relation which expresses the resulting moment τ_p at the ZMP, as a function of the physical quantities that characterize the robot motion (introduced in section 2.6.1), as follows:

$$\tau_p = \dot{\mathcal{L}} - c \times MG + (\dot{\mathcal{P}} - MG) \times p \quad (2.117)$$

As regarded in equation (2.109), τ_p is a tridimensional vector whose first two elements are null, since the components of the resulting moment of ground reaction forces expressed at the ZMP are equal to zero at the horizontal plane; therefore, from equation (2.117) above, the horizontal components of the resulting moment τ_p can be rewritten as:

$$\tau_{p_x} = \dot{\mathcal{L}}_x + Mgy + \dot{\mathcal{P}}_y p_z - (\dot{\mathcal{P}}_z + Mg)p_y = 0 \quad (2.118)$$

$$\tau_{p_y} = \dot{\mathcal{L}}_y - Mgx - \dot{\mathcal{P}}_x p_z + (\dot{\mathcal{P}}_z + Mg)p_x = 0 \quad (2.119)$$

where

$$\dot{\mathcal{P}} = \begin{pmatrix} \dot{\mathcal{P}}_x \\ \dot{\mathcal{P}}_y \\ \dot{\mathcal{P}}_z \end{pmatrix}, \quad \dot{\mathcal{L}} = \begin{pmatrix} \dot{\mathcal{L}}_x \\ \dot{\mathcal{L}}_y \\ \dot{\mathcal{L}}_z \end{pmatrix}, \quad c = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{and} \quad G = \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix}$$

Finally, the resolution of the system obtained from equation (2.118) and (2.119) provides the values for p_x and p_y , describing the calculation of the ZMP position based on the robot movements and are given by:

$$p_x = \frac{Mgx + p_z \dot{\mathcal{P}}_x - \dot{\mathcal{L}}_y}{Mg + \dot{\mathcal{P}}_z} \quad (2.120)$$

$$p_y = \frac{Mgy + p_z \dot{\mathcal{P}}_y + \dot{\mathcal{L}}_x}{Mg + \dot{\mathcal{P}}_z} \quad (2.121)$$

It is possible to identify from the previous relations that, in the special case the robot is in a immobile condition, all components of the linear and angular momentums are equal to zero and therefore, $p_x = x$ and $p_y = y$; this means that in this situation, the position of the ZMP coincide with the position of the CoM projection to the ground, as previously established in section 2.6.2.

2.7 Walking Pattern Generation

From the viewpoint of control and walking pattern generation the works developed by the researchers could be classified into two categories. The first category, which is the most commonly used, requires the precise knowledge of robot dynamics (e.g., mass, location of the CoM, inertial tensor, linear and angular momentums, etc.) of each model body, in order to prepare efficient walking patterns. Therefore, it mainly relies on the accuracy of the models. This group is called the ZMP based approach since they often use the zero-moment point (ZMP) for pattern generation and walking control.

In contrast, there is the second category to generate walking patterns which uses limited knowledge of dynamics. Since the controller knows little about the system structure, this approach much relies on feedback control. This is called the inverted pendulum approach, since they frequently uses a tridimensional linear inverted pendulum model (3D-LIP).

From the second standpoint, there are some works like [18] and [19] where the authors proposed walking control and pattern generation, by which dynamic biped walking was successfully realized on simulations and experiments. However, this method was not applicable to a situation like a walking on stepping-stones where the foot must be placed on a specific location. Most of the inverted pendulum based methods suffer with this problem while the ZMP based methods like [20] can handle such situation.

The following sections describe the general nature of these two mentioned approaches for walking pattern generation and finally a useful scheme based on the preview control of the ZMP, according to the work developed in [1] is discussed.

2.7.1 Generation of a walking pattern based on the 3D-LIP

The first method to describe both human and robotic biped locomotion is based on a three dimensional linear inverted pendulum model (3D-LIP), as shown in Figure 2.37. This model consists of a center of mass (CoM) which is linked to a support point by a telescopic leg without mass.

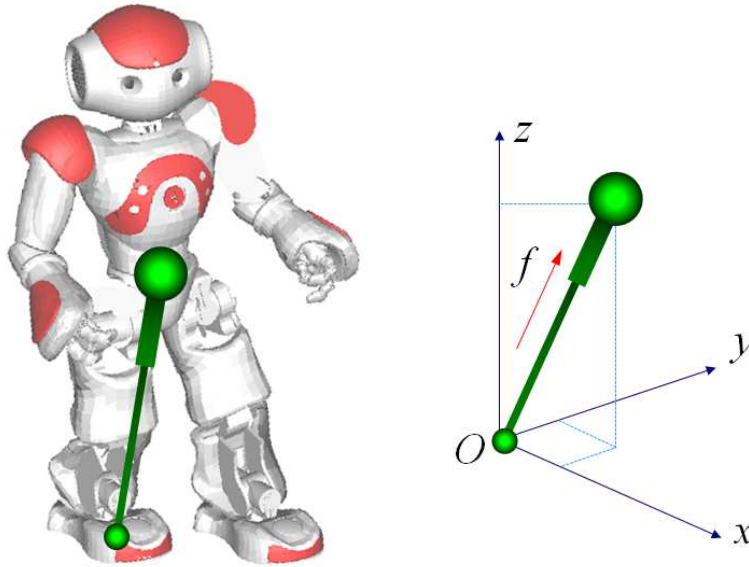


Figure 2.37 - Model of a walking robot based on the 3D-LIP

By applying a constraint control to an inverted pendulum such that a mass m should move along an arbitrary defined plane, as shown in Figure 2.38, a simple linear dynamics model is obtained; this model is called the **Tridimensional Linear Inverted Pendulum Model (3D-LIP)** [21]. For standardization purposes, Cartesian coordinates are used and the x -axis is specified as the ordinal walking direction. The constraint plane is represented with given normal vector $(k_x, k_y, -1)$ and z intersection z_c as:

$$z = k_x x + k_y y + z_c \quad (2.122)$$

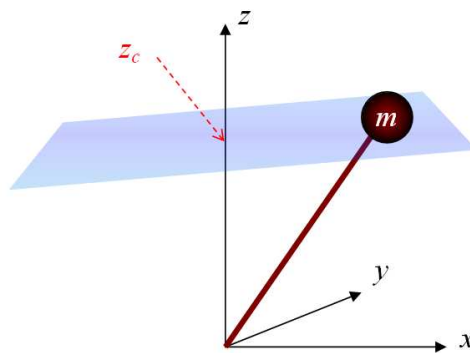


Figure 2.38 - 3D inverted pendulum under constraint

If the constraint plane is horizontal ($k_x = k_y = 0$), the dynamics under the constraint control is given by:

$$\ddot{x} = \frac{g}{z_c} x - \frac{1}{mz_c} \tau_y \quad (2.123)$$

$$\ddot{y} = \frac{g}{z_c} y - \frac{1}{mz_c} \tau_x \quad (2.124)$$

where m is the mass of the pendulum, g is acceleration of gravity, τ_x and τ_y are the moments around z -axis and y -axis respectively. Even in the case of a sloped constraint, where $k_x, k_y \neq 0$, the same dynamics are obtained by applying an additional constraint for the input torques:

$$\tau_x x + \tau_y y = 0 \quad (2.125)$$

Equations (2.123) and (2.124) are linear equations. The only parameter which governs those dynamics is z_c , i.e., the z intersection of the constraint plane and the inclination of the plane never affects the horizontal motion.

For the 3D-LIPM considering the horizontal constraint ($k_x = k_y = 0$), the position of the **Zero-Moment Point (ZMP)** can be easily calculated, according to [1], by using the relations:

$$p_x = -\frac{\tau_y}{mg} \quad (2.126)$$

$$p_y = +\frac{\tau_x}{mg} \quad (2.127)$$

where (p_x, p_y) represents the location of the ZMP on the ground. By solving for the moments in equations (2.126) and (2.127) and substituting on the 3D-LIP model in equations (2.123) and (2.124), it is possible to obtain the equations of dynamics of a 3D-LIP with the supporting point fixed at the origin, as follows.

$$\ddot{x} = \frac{g}{z_c} (x - p_x) \quad (2.128)$$

$$\ddot{y} = \frac{g}{z_c} (y - p_y) \quad (2.129)$$

The analytic solution to the differential equation of dynamics (2.128), representing the displacement in x direction is given by the following relations:

$$x(t) = (x(0) - p_x) \cosh\left(\frac{t}{T_c}\right) + T_c \dot{x}(0) \sinh\left(\frac{t}{T_c}\right) + p_x \quad (2.130)$$

$$\dot{x}(t) = \frac{x(0) - p_x}{T_c} \sinh\left(\frac{t}{T_c}\right) + \dot{x}(0) \cosh\left(\frac{t}{T_c}\right) \quad (2.131)$$

where $T_c = \sqrt{\frac{z_c}{g}}$. For the displacement in y direction, the solution is obtained identically.

Considering this behavior, it is possible to design a walking model based on the 3D-LIP, where a trajectory, like the one shown in Figure 2.39(a), is generated by assembling a series of walking primitives, which are portions of symmetric hyperbole, defined between the time interval $0 \leq t \leq T_{\text{sup}}$, where T_{sup} is the supporting time of each step, as shown in figure 5.16(b).

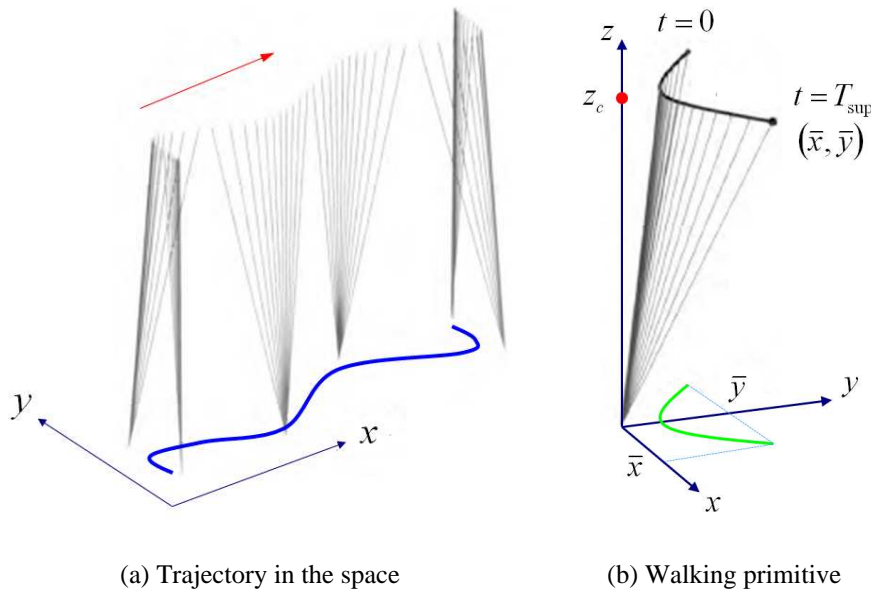


Figure 2.39 - Generation of a walking pattern based on the 3D-LIP

2.7.2 Generation of a walking pattern based on the ZMP.

Another method for walking pattern generation is the **cart-table model**, shown in figure 5.17. This is the most representative of the ZMP based models. It consists of a cart of mass M that moves on a table of negligible mass. The foot of the table is very small to preserve static equilibrium, since the chart is very close to the platform edge; nevertheless it is possible to maintain dynamic equilibrium, by applying a sufficient acceleration to the cart. This model considers a unique mass at a constant height.

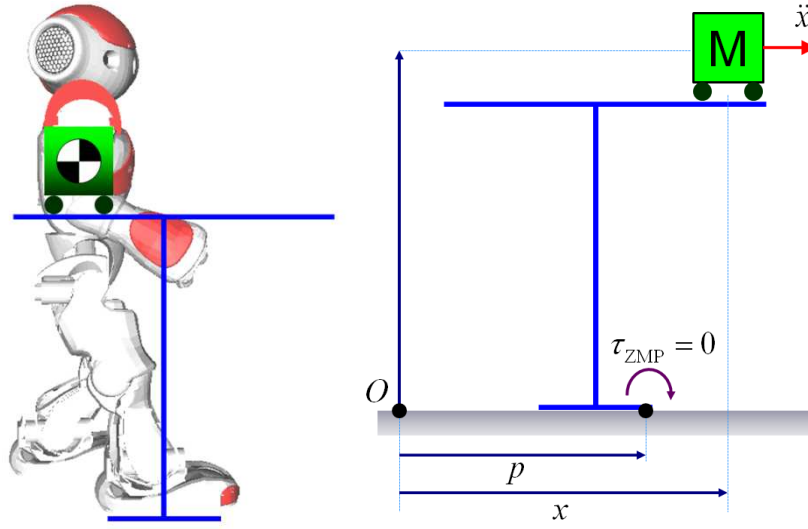


Figure 2.40 - Cart-table model. The dynamics of the walking robot is modeled by a cart that moves over a mass-less table. The state of the moving determines the position of the ZMP.

By regarding equations (2.128) and (2.129), and then solving for p_x and p_y , respectively, it is possible to obtain the equations which determine the position of the ZMP, according to the cart-table model, as follows:

$$p_x = x - \frac{z_c}{g} \ddot{x} \quad (2.132)$$

$$p_y = y - \frac{z_c}{g} \ddot{y} \quad (2.133)$$

It is evident that the two previous equations, also called **ZMP equations** and equations (2.128) and (2.129), which describe the dynamics of the 3D-LIP, are the same, but with different outputs. This is the main difference between the two models of walking pattern generation. In the 3D-LIP model the trajectory of the CoM is generated from the desired position of the ZMP; oppositely, in the cart-table model, the position of the ZMP is generated from the desired trajectory of the CoM. The relationship between the two models is the inversion of the inputs and outputs.

In the case of the 3D-LIP model, it is sometimes difficult to execute a desired ZMP trajectory, as the support foot position (ZMP) should be modified in certain steps during the walking transition. In the cart-table model, the desired ZMP trajectory is realized and therefore is the most suitable for walking pattern generation.

By representing a robot as the cart-table model and by considering the cart motion as the trajectory of the whole robot CoM, the resulted ZMP can be easily calculated by using the ZMP equations (2.132) and (2.133). Nonetheless, a walking pattern generation is the

inverse problem of this; this means that the cart motion should be calculated from the given ZMP trajectory, which is determined by the desired footholds and step period.

Takanishi et al. in [22] proposed to solve this problem by applying the Fast Fourier Transformation (FFT) to the ZMP reference, the ZMP equations can be solved in frequency domain. Then the inverse FFT returns the resulted CoM trajectory into time domain. Besides, Kagami, Nishiwaki et al. in [20] proposed a method to solve this problem in the discrete time domain. They showed that the ZMP equation can be discretized as a trinomial expression, and it can be efficiently solved by an algorithm of $O(N)$ for the given reference data of size N .

Both methods are proposed as batch processes that use a ZMP reference of certain period and generate the corresponding CoM trajectory. To generate continuous walking pattern for a long period, they must calculate entire trajectory by off-line or must connect the piece of trajectories calculated from the ZMP reference divided into short segments.

ZMP Control as a servo problem

In order to represent the ZMP control as a servo problem, it is first necessary to define a variable u_x as the time derivative of the acceleration¹⁰, thus:

$$u_x = \frac{d}{dt} \ddot{x} = \dddot{x} \quad (2.134)$$

Now, regarding u_x as the input of equation (2.132), it is possible to translate the ZMP equation into a strictly proper dynamical system as:

$$\begin{aligned} d \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_x \\ p_x &= \begin{pmatrix} 1 & 0 & \frac{-z_c}{g} \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} \end{aligned} \quad (2.135)$$

For the movement in y direction, the system is obtained in the same form.

Finally, by using the dynamics of system (2.135), it is possible to construct a walking pattern generator as a ZMP tracking control system, as shown in Figure 2.41. The system generates the CoM trajectory such that the resulted ZMP follows the given reference.

¹⁰ The time derivative of the acceleration ($\frac{d}{dt} \ddot{x}$) is also denominated *jerk*.

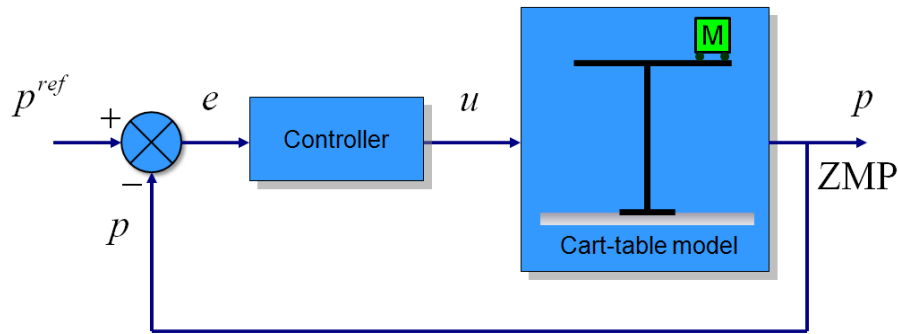


Figure 2.41 - Control system used to follow the reference position of the ZMP

2.7.3 Walking Pattern Generation based on the Preview Control of the ZMP

The control loop described in the previous section presents an interesting feature, which must be considered. Figure 2.42 illustrates an example of the ideal trajectories of the ZMP and the CoM of a robot that walks one step forward dynamically. The robot supports its body by hind-leg from 0s to 1.5s, and has support exchange at 1.5s followed by the foreleg support until 3.0s. Thus the reference ZMP should have a step change at 1.5s and obviously the CoM must start moving **before** this. The use of a standard controller, such as the one shown in Figure 2.41, will not provide a proper walking pattern, since in an ordinary servo-system, the desired output is obtained with a time delay, respect to the change in the reference signal. Therefore, in order to generate a walking pattern for biped robots, the output signal must be calculated from the **future** input.

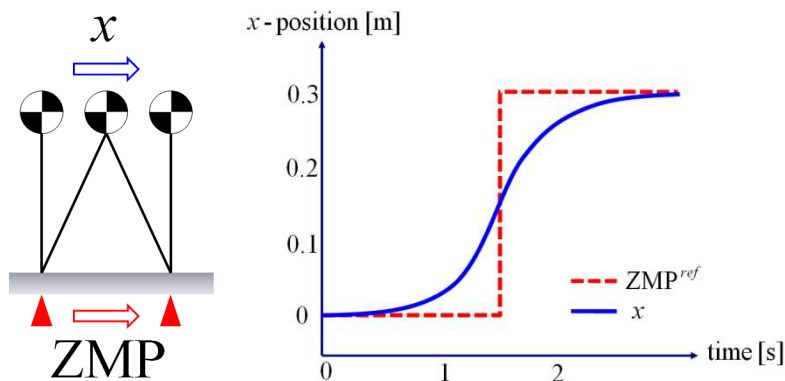


Figure 2.42 - Ideal trajectory of the ZMP and the CoM for one step forward

A control method that is based on the future information is called **preview control**, according to [23]. Particularly, an optimal preview servo control method was proposed in

1985 by Kayatama, et al. in [24]. This preview control method was applied for humanoid walking pattern generation by Kajita et al. in [1].

In order to describe this preview control method it is necessary, in first place, to discretize the system (2.135) with a fixed sampling time T as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ p(k) &= Cx(k) \end{aligned} \quad (2.136)$$

where

$$x(k) = \begin{pmatrix} x(kT) \\ \dot{x}(kT) \\ \ddot{x}(kT) \end{pmatrix} \quad u(k) = u_x(kT) \quad p(k) = p_x(kT)$$

and

$$A = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} T^3/3 \\ T^2/2 \\ T \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 & -z_c/g \end{pmatrix}$$

According to system (2.136), by the knowledge of the current value of the state vector $x(k)$ and the calculation of the control variable $u(k)$ it is possible to determine the subsequent value of the state vector $x(k+1)$, considering that the constant parameters T , z_c and g are given as input information. Thus, by applying this formulation for each sample, a complete trajectory of the CoM can be constructed from the input reference of ZMP position; only the initial conditions of the state vector $x(0)$ are needed. In such way, it is possible to generate a discrete-time walking pattern based on the cart-table model.

By the incorporation of a preview controller, the standard servo-system shown in Figure 2.41 is modified by the control scheme shown in Figure 2.43, where it is possible to identify that at sample k , the value of the control variable $u(k)$ depends on the state vector $x(k)$, on the error signal $e(k)$ and also on a finite amount N_L of future reference samples of the ZMP position $\{p^{ref}(k+1), \dots, p^{ref}(k+N)\}$.

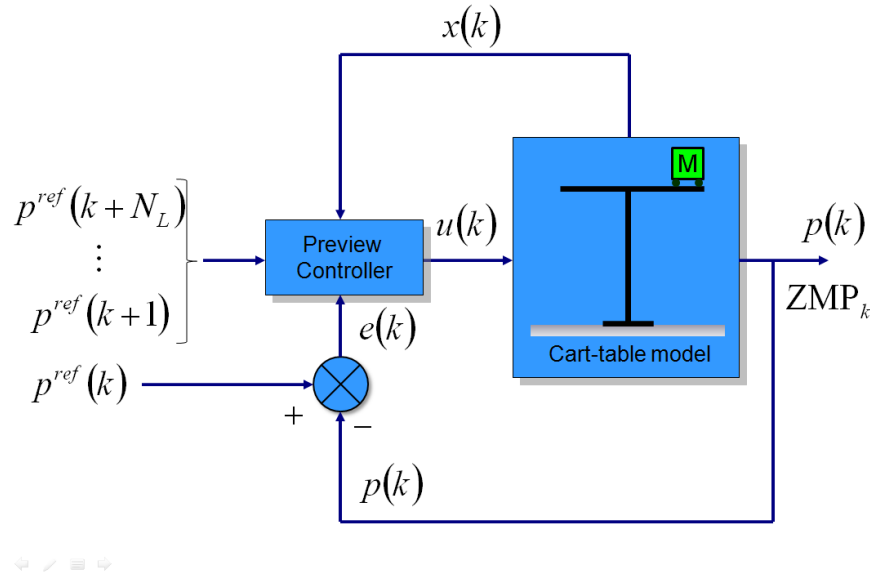


Figure 2.43 – Control system using future reference of ZMP position

Besides, according to [24], in order to make the output ZMP position $p(k)$ follow the reference ZMP position $p^{ref}(k)$ the closest as possible, it is necessary to consider the following minimization problem

$$J = \sum_{i=k}^{\infty} [e^T(i)Q_e e(i) + \Delta x^T(i)Q_x \Delta x(i) + \Delta u^T(i)R \Delta u(i)] \quad (2.137)$$

where $e(j) = p(j) - p^{ref}(j)$ is the error signal, $Q_e, R > 0$ and Q_x is a 3 x 3 symmetric non-negative definite matrix. $\Delta x(k) = x(k) - x(k-1)$ is the incremental state vector and $\Delta u(k) = u(k) - u(k-1)$ is the incremental control variable.

When the ZMP reference can be previewed for N_L future steps respect to a determined sample k , the optimal control variable $u(k)$ which minimizes the performance index J of problem (2.137) is given by the following expression:

$$u(k) = -G_I \sum_{i=0}^k e(i) - G_x x(k) - \sum_{j=1}^{N_L} G_d(j) p^{ref}(k+j) \quad (2.138)$$

The preview control of equation (2.53) consists of three terms, the integral action on the tracking error, the state feedback and the preview action using the future reference. The constants G_I , G_x and $G_d(j)$ are the control gains calculated from the weights Q_e , Q_x , R and the coefficients A , B and C of system (2.136), and are given by:

$$G_I = [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{I} \quad (2.139)$$

$$G_x = [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{F} \quad (2.140)$$

$$\begin{aligned} G_d(1) &= -G_l \\ G_d(l) &= [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{X}(l-1) \end{aligned} \quad (2.141)$$

where

$$\tilde{B} = \begin{bmatrix} CB \\ B \end{bmatrix}, \quad \tilde{I} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \tilde{F} = \begin{bmatrix} CA \\ A \end{bmatrix}, \quad \tilde{Q} = \begin{bmatrix} Q_e & 0 \\ 0 & Q_x \end{bmatrix}, \quad \tilde{A} = \begin{bmatrix} \tilde{I} & \tilde{F} \end{bmatrix} \quad (2.142)$$

and where the matrices $\tilde{X}(l)$ are given by

$$\tilde{X}(l) = \tilde{A}_c^T \tilde{X}(l-1), \quad l = 2, \dots, N_L; \quad \tilde{X}(1) = -\tilde{A}_c^T \tilde{K} \tilde{I} \quad (2.143)$$

where \tilde{A}_c is the closed-loop matrix defined by

$$\tilde{A}_c = \tilde{A} - B [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{A} \quad (2.144)$$

Furthermore, the matrix \tilde{K} is the non-negative definite solution of the discrete-time algebraic Riccati equation

$$\tilde{K} = \tilde{A}^T \tilde{K} \tilde{A} - \tilde{A}^T \tilde{K} \tilde{B} [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{A} + \tilde{Q} \quad (2.145)$$

which can be computed by using different methods. A useful technique to solve this equation was proposed by Laub in [25] and uses the RSF representation of a symplectic matrix constructed from the Riccati equation parameters. A complete explanation of this method is included in Appendix D.

The control Gain $G_d(j)$ emphasizes the weight of the nearest future reference and minimizes the weight of the more distant future reference, as shown in Figure 2.44; Kajita et. al in [1] propose to use a future reference ($T \cdot N_L$) of 1.6 seconds in order to provide a proper system output.

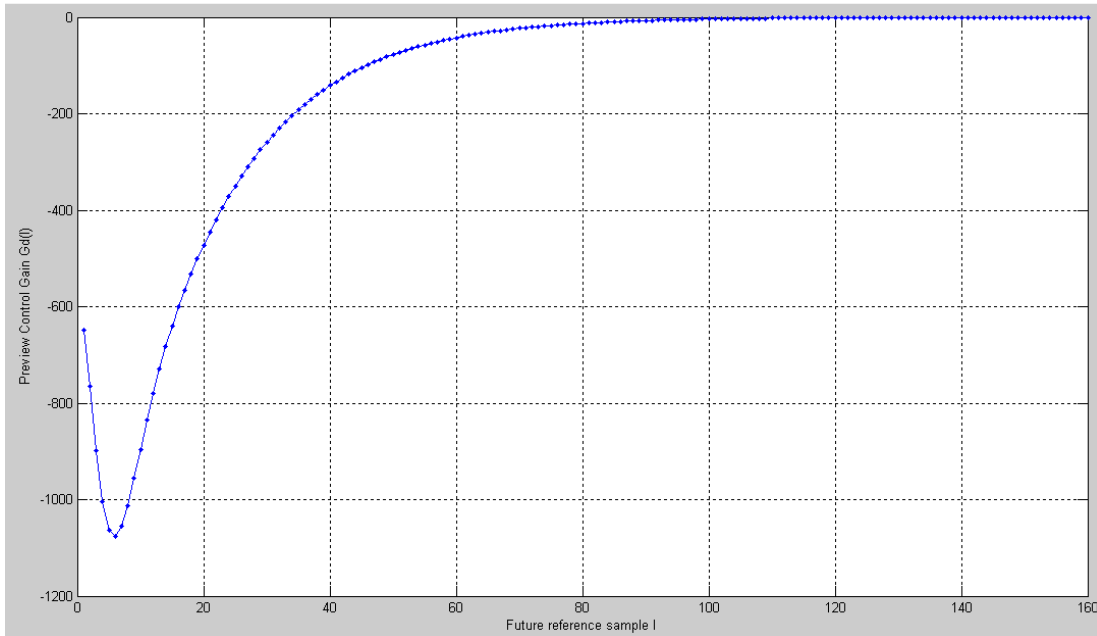


Figure 2.44 – Value of the preview control gain $G_d(l)$, considering $N_L = 160$, $T = 1 \times 10^{-3}$ s ,
 $Q_e = 1$, $Q_x = 0$ and $R = 1 \times 10^{-6}$.

Chapter 3

Methodology

The main objective of this chapter is on one hand, to approach from a more practical perspective a series of relevant aspects that were considered for the application of all theoretic fundamentals introduced in Chapter 2 in order to design an efficient walking pattern generator, suitable for visually guided tasks. On the other hand, a series of simulations in computer, as well as experiments in a physical platform were performed, with the intention of validating the designed systems from both kinematic and dynamic viewpoints.

Firstly, the strategies that were followed through the development of the present Master thesis project are exposed in section 3.1. By its part, section 3.2 goes deeper into the inverse kinematics scheme that was explained in section 2.5.6 and clarifies the way to represent a series of kinematic tasks (that are used for practical locomotion purposes) as linear systems of equalities or inequalities and consequently construct the stack of tasks that is to be introduced to the PIK algorithm.

Moreover, a global method to design a walking pattern generator is presented in section 3.3, considering two stages of preview control of ZMP and PIK, with the aim of providing a sequence of articular configurations, from the reference position of the ZMP.

Finally, since there is no documental or experimental information available about the optimal leg trajectory planning, in section 3.4 the strategy that was applied in this thesis work to attain this goal is proposed.

3.1 Development strategies

It is important to mention that this work did not start from scratch. Several software modules and useful bibliography material have been provided by the supervisor in order to facilitate the project development.

The point of this master thesis work has been to grasp into the state-of-the art and, if necessary, some other material and to make a practical implementation of the generated system in a physical humanoid platform.

In order to clarify certain technical points about the fundamentals and/or the implementation of the algorithms, it was possible to contact and to meet some of the authors of the referenced papers via the supervisor; furthermore, regular discussions during the thesis work were programmed.

Once a month new pieces of work were shown, as well as some sketches of preliminary solutions or representative examples of the proposed ideas.

The followed methodology has been:

- Develop the necessary analytical and numerical tools for implementing the walking pattern generator.
- Implement the solution both in simulation and in the experimental robotic platforms.
- Perform exhaustive sensitivity analysis of the numerical implementation.
- Design several scenarios to validate the algorithms.

In order to attain the project objectives, the following equipment and computational tools have been utilized:

- Humanoid Robot NAO, developed by Aldebaran Robotics [26].
- High speed computer.
- Mathematical calculations and simulation software MATLAB® 2007.
- Microsoft® Visual Studio 2008 for Object Oriented Programming C/C++.
- *Naoqi* module management interface for NAO.
- Simulation and communication interface *Choregraphe* for NAO.
- Programming language Python for Linux.
- NAO VRML model.
- BLAS/LAPACK linear algebra module libraries [27].

A detailed timeline with projected and real dates of the developed activities to fulfill the present master thesis project requirements is shown in Table 3-1.

Activity	R/P	Start date	End date
Bibliography review, problem formulation and construction of preliminary examples of the walking pattern generator based on the preview control of the ZMP	Projected	07.03.2011	01.04.2011
	Real	7.03.2011	15.04.2011
Basic analytical and numerical tools development.	Projected	04.04.2011	29.04.2011
	Real	18.04.2011	29.04.2011
Solution implementation by using Object Oriented Programming.	Projected	2.05.2011	3.06.2011
	Real	2.05.2011	24.06.2011
Algorithms validation in simulation interfaces.	Projected	6.06.2011	1.07.2011
	Real	27.06.2011	22.07.2011
Whole strategy validation in a real humanoid robot (NAO)	Projected	4.07.2011	5.08.2011
	Real	25.07.2011	12.08.2011
Thesis redaction	Projected	8.08.2011	2.09.2011
	Real	16.06.2011	18.08.2011

3.2 Mathematical representation of kinematic tasks

As approached in section 2.5.6, all kinematic tasks for a robot manipulator can be represented as linear systems of equalities or inequalities, in accordance with their specific nature. This mathematical representation is subsequently used to compute the inverse kinematics by using an adequate solution strategy as proposed in Algorithm 2.4 (in the case of having only equality constraints), and in Algorithm 2.5 (when considering mixed equality and inequality constraints).

The following subsections explain the mathematical foundation that is utilized to represent a series of kinematic tasks which are used in humanoid locomotion, giving a particular emphasis in biped walking purposes.

3.2.1 Position and Orientation

Perhaps the most common kinematic task, not only for humanoid robots, but also for any kind of manipulator is to locate a specific frame that is associated to a body of the robot model, at a desired position and orientation in the 3D space, respect to the world reference frame. In humanoid robots, this task is mainly used, for instance, to place the feet on the floor and also for reaching/grasping applications. A kinematic task of this nature is represented as an equality constraint, in the form:

$$J_e(q)\dot{q} = \dot{e}_e(q) \quad (3.1)$$

where q is the articular variable in the Configuration Space, $J_e(q)$ is the Jacobian matrix associated to the point on the robot model that performs the task, and where $e_e(q)$ is the task error between the desired and the current position/orientation of the considered point, which is given by

$$e_e(q) = x_{des} - x(q) = \begin{pmatrix} p_{des} \\ \alpha_{des} \end{pmatrix} - \begin{pmatrix} p \\ \alpha \end{pmatrix} = \begin{pmatrix} x_{des} \\ y_{des} \\ z_{des} \\ \psi_{des} \\ \theta_{des} \\ \phi_{des} \end{pmatrix} - \begin{pmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \phi \end{pmatrix} \quad (3.2)$$

3.2.2 Articular Limits

Regardless of the main purpose the robot motion is destined to, articular limits must be always considered during the inverse kinematics procedure. This task might be included as a clamping routine, as in Algorithm 2.4, or otherwise it can be represented as a double inequality system in the following way:

$$q_{low} \leq q \leq q_{up} \quad (3.3)$$

which is transported to the velocity space as follows

$$q_{low} - q \leq \dot{q} \leq q_{up} - q \quad (3.4)$$

In order to represent the previous inequality in the same form as in system (2.75), it should be separated into two individual systems of linear inequalities as:

$$\begin{aligned} I \dot{q} &\leq q_{up} - q \\ -I \dot{q} &\leq q - q_{low} \end{aligned} \quad (3.5)$$

where I is an $n \times n$ identity matrix, considering that n is the total amount of robot DoF.

3.2.3 Obstacle Avoidance

In several locomotion applications, there are obstacles which should be avoided. These obstacles might be external points respect to the robot model (which can be fixed or mobile in space) or might be also points inside the kinematic model (i.e., self collision avoidance).

A formulation for obstacle avoidance is proposed in [28] and considers the distance d between the two closest points p_1 and p_2 belonging to two objects O_1 and O_2 , respectively, as illustrated in Figure 3.1. An inequality constraint is constructed, as a function of the speed of distance d as follows:

$$-\dot{d} \leq \xi \frac{d - d_s}{d_i - d_s} \quad (d_s < d_i) \quad (3.6)$$

where d_i is an influence distance, d_s is a security distance and ξ is a positive coefficient used to adjust the convergence velocity.

The speed \dot{d} is a scalar value which is given by the following inner product:

$$\dot{d} = p_1 \bullet n \quad (3.7)$$

where $n = (p_1 - p_2)/d$ is a unitary vector describing the direction of the line passing through points p_1 and p_2 . By considering that O_1 corresponds to a robot body, and that O_2 is an obstacle (or other robot body) to be avoided, then \dot{p}_1 can be expressed as a function of the articular configuration q and it is given by:

$$\dot{p}_1(q) = J_1(q)\dot{q} \quad (3.8)$$

where $J_1(q)$ is the Jacobian matrix associated to the inequality task (i.e., the point p_1 that is associated to the body O_1). Finally, the inequality constraint that represents the obstacle avoidance task can be expressed as follows:

$$-n^T J_1(q)\dot{q} \leq \xi \frac{d - d_s}{d_i - d_s} \quad (3.9)$$

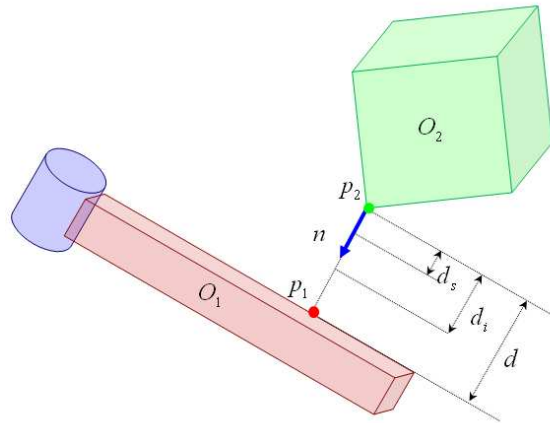


Figure 3.1 – Method to represent obstacle avoidance as a linear system of inequalities

3.2.4 Static Balance

When a humanoid robot is executing a task (or a set of tasks), it is important to control its balance with the aim of ensuring that its motion is dynamically feasible (i.e., the robot will not fall) and therefore, static balance should be also considered as a kinematic task.

As approached in section 2.6.2, static balance can be guaranteed by maintaining the projection of the robot CoM inside the support polygon, as shown in Figure 3.2. Consider O as the support polygon centroid, C as the CoM projection to the ground and P as the intersection point of the line passing through O , C and in the support polygon edge;

furthermore, d_{OC} is the distance from O to C , d_{OP} is the distance from O to P and d_s is a security distance. Thus, static balance can be mathematically represented by the following inequality constraint:

$$d_{OC} \leq d_{OP} - d_s \quad (3.10)$$

which transported to the velocity domain, can be expressed as:

$$\dot{d}_{OC} \leq d_{OP} - d_s - d_{OC} \quad (3.11)$$

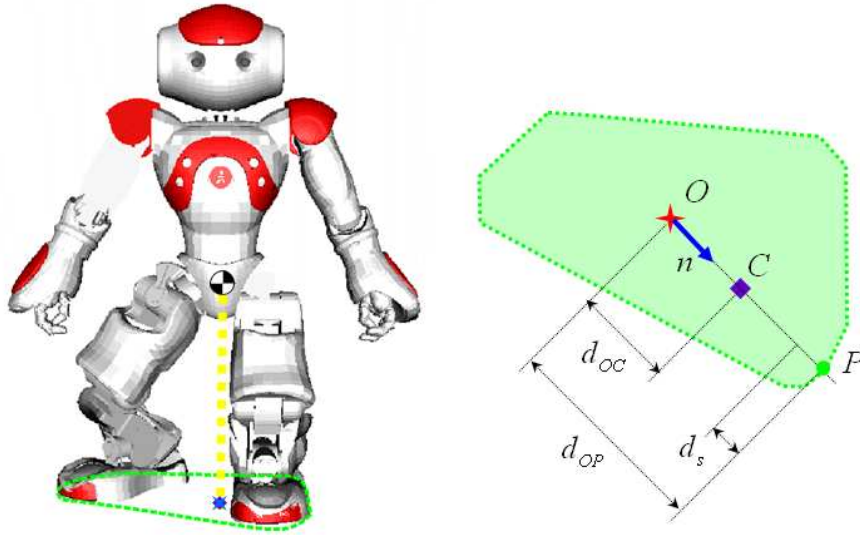


Figure 3.2 – CoM projection to the ground maintained inside the support polygon surface

Moreover, being d_{OC} dependent of the articular configuration q , then the previous constraint can be rewritten as follows:

$$nJ_{CoM}(q) \leq d_{OP} - d_s - d_{OC} \quad (3.12)$$

where n is a unitary vector describing the direction of segment \overline{OC} and $J_{CoM}(q)$ is the whole body CoM Jacobian matrix¹¹, which according to [29] its geometric version can be determined as a weighted average of all model bodies CoM geometric Jacobian matrices, as follows:

¹¹ When the locomotion application considers a non-inertial base, then $J_{CoM}(q)$ should be expressed in its mobile manipulator Jacobian version $mmJ_{GCoM}(q)$

$$J_{GCoM} = \frac{\sum_{i=0}^{N-1} m_i J_{G_i}}{\sum_{i=0}^{N-1} m_i} \quad (3.13)$$

Where N is the total amount of bodies constituting the robot model, m_i is the mass of body i , and J_{G_i} is the geometric Jacobian matrix of the CoM of body i , calculated as proposed in Algorithm 2.2.

3.3 Global Architecture of the Walking Pattern Generator

This section grasps directly into the objectives of the present project and particularly, enlightens the details about the design of a complete walking pattern generator for a biped humanoid robot which allows non-holonomic locomotion for visually guided tasks. The proposed walking pattern generator fulfills the following requirements:

- Generate a two-dimensional trajectory for the humanoid CoM, from a given reference position of the ZMP, which guarantees that the locomotion is dynamically stable.
- Generate a three-dimensional trajectory for both robot feet, from the given reference position of the ZMP.
- Apply a kinematic control scheme which allows that the robot head will be permanently aiming at a fixed landmark point during the whole locomotion.
- Provide a set of articular position values (vector q) which describe, at each instant sample, the root configuration during the whole trajectory.

With the intention of designing this walking pattern generator, the fundamentals approached in Chapter 2 must be taken together into a **global architecture** having the following input/output parameters outlined in Table 3-2.

Table 3-2 Inputs and output of the Global Architecture for Walking Pattern Generation			
	Description	Representation	Unit
Inputs	Reference ZMP position for each footstep (n)	$p_n^{ref} = (p_{x_n}^{ref} \quad p_{y_n}^{ref} \quad p_{z_n}^{ref})^T$	[m]
	Single support time	T_{ss}	[sec]
	Double support time	T_{ss}	[sec]
	CoM z-position	z_c	[m]
	ZMP z-position	p_z	[m]
	Sampling time	T	[sec]
	Fixed landmark position	$L = (L_x \quad L_y \quad L_z)^T$	[m]
Output	Articular configuration vector at each sample (k) for the N joints of the kinematic model	$q_k = (q_{1k} \quad \dots \quad q_{Nk})^T$	[rad]

The first step is to represent the **ZMP reference position**, which is originally parameterized respect to the footsteps domain $n = \{1, \dots, n_{max}\}$, into a **discrete-time parameterized input variable**, described in terms of k . The total samples per footstep is

given by $(T_{ss} + T_{ds})/T$; therefore, a single footstep is represented in the discrete time by considering the interval

$$n(T_{ss} + T_{ds}) \leq kT \leq (n+1)(T_{ss} + T_{ds})$$

Thus, the transition on x and y directions of the ZMP reference position from footstep $n-1$ to n , and subsequently from footstep n to $n+1$, has a behavior in the discrete-time domain such as the one illustrated in Figure 3.3.

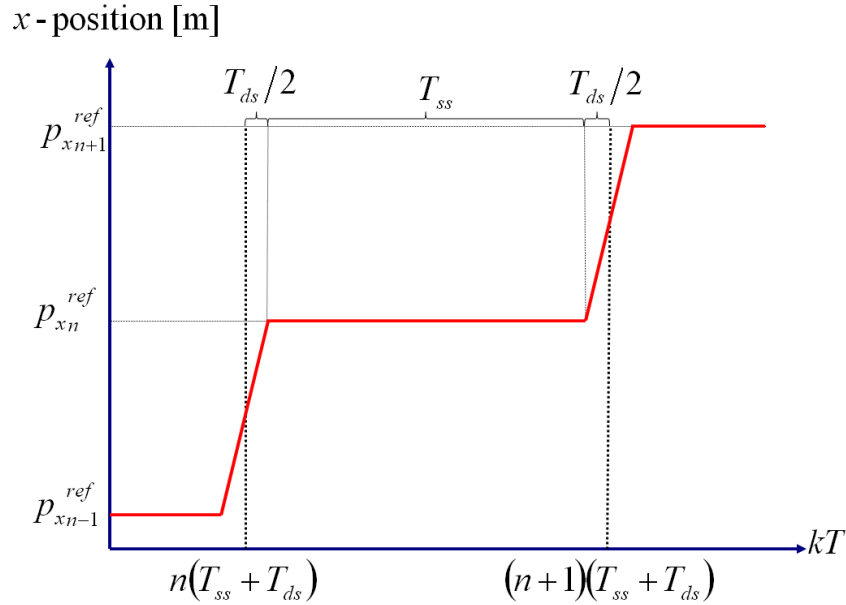


Figure 3.3 – Transition of the ZMP x -position among subsequent footsteps, represented as a discrete-time parameterized variable. Behavior in y -direction has an identical scheme

Once having the discrete-time representation of the ZMP reference position, the following step is to **incorporate the linear-quadratic regulator (LQR)** explained in section 2.7.3 and illustrated in Figure 2.43, for each direction x and y and considering the state vector as the block output. The LQR has then following transfer function:

$$c_{k+1}^* = f(p_k^{ref}, p_{k+1}^{ref}, \dots, p_{k+N_L}^{ref}) \quad (3.14)$$

where $c_{k+1}^* = (x_{k+1}^* \quad y_{k+1}^* \quad z_c)$ is the CoM position for the subsequent sample $k+1$. Figure 3.4 illustrates the LQR as a system block

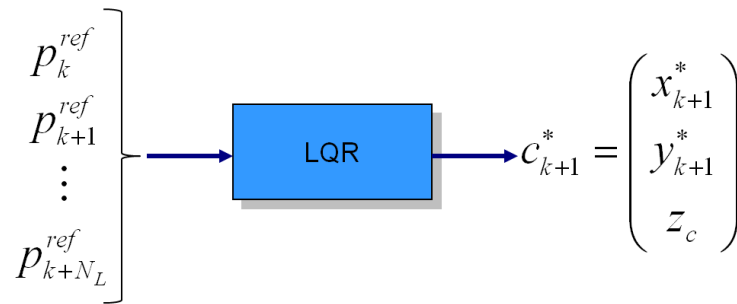


Figure 3.4 – Linear Quadratic Regulator representing the Preview Control of the ZMP used to determine the trajectory of the robot CoM

The next step in the design of the complete walking pattern generator for visually guided tasks is to construct a **stack of kinematic tasks** (SoT) which should include for a considered sample $k+1$, the following:

- Desired CoM position c_{k+1}^*
- Both feet desired position and orientation for sample $k+1$.
- Head desired orientation for sample $k+1$.

At this point, the robot CoM position for sample $k+1$ has been already determined at the LQR stage; moreover, section 3.4 proposes methods to determine the feet trajectories and the desired head orientation.

The constructed SoT is submitted to a **Prioritized Inverse Kinematics (PIK)** stage which, as studied in section 2.5.6, provides the following **articular configuration** q_{k+1} that fulfills, with a decreasing priority order, the established desired positions and/or orientations.

So far, a resulting sequence of articular configurations has been obtained from the PIK stage, which describes the robot locomotion; with this information it is possible to determine the articular velocity \dot{q}_k by using numeric differentiation of the articular values. From this information it is possible to determine each body linear and angular velocities (equations (2.16) and (2.17), respectively), as well as the whole model linear and angular momentums (equations (2.84) and (2.85), respectively).

The previous parameters are computed in order to **approximate the real position of the ZMP** which should be also calculated, based on the robot movements. This is done by using equations (2.120) and (2.121), where parameters \dot{P} and \dot{L} can be calculated by using numeric differentiation of the whole model linear and angular momentum, respectively. A block with the cited operations so far is presented in Figure 3.5

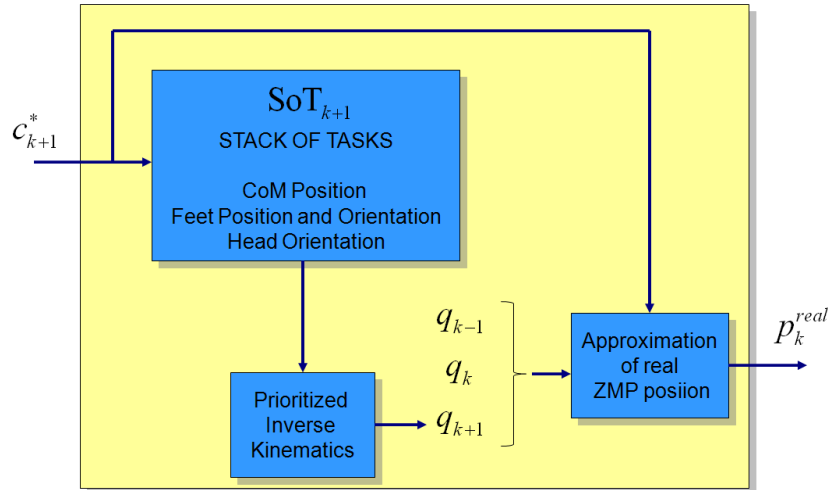


Figure 3.5 – Execution of the PIK stage and approximation of the real ZMP position

The purpose of calculating an approximation of the real ZMP position at each sample k is to utilize these measurements in a second systematic stage, proposed in [1], which also incorporates a LQR, identical to the first one, considering N_L samples of deviation between the reference and real ZMP positions ($\Delta p_k^{ref} = p_k^{ref} - p_k^{real}$). The output of this LQR provides an optimal CoM position correction (Δc_k^*) which is to be added to the CoM position (c_k^*) obtained from the first stage, in order to minimize the ZMP deviation attributable to the model simplifications.

The corrected CoM position ($c_k^* + \Delta c_k^*$), as well as the feet position/orientation and the head orientation tasks, are also taken together into a SoT and submitted to a second PIK process which provides the optimal articular configuration (q_k^*) considering the ZMP position correction.

By assembling both operation stages then a **global architecture of the Walking Pattern Generator** is constructed which, as shown in Figure 3.6, supplies at instant sample k the subsequent optimal articular configuration q_{k+1}^* to attain the kinematic tasks desired values, from the input ZMP reference position of the adjacent $2N_L - 1$ future samples.

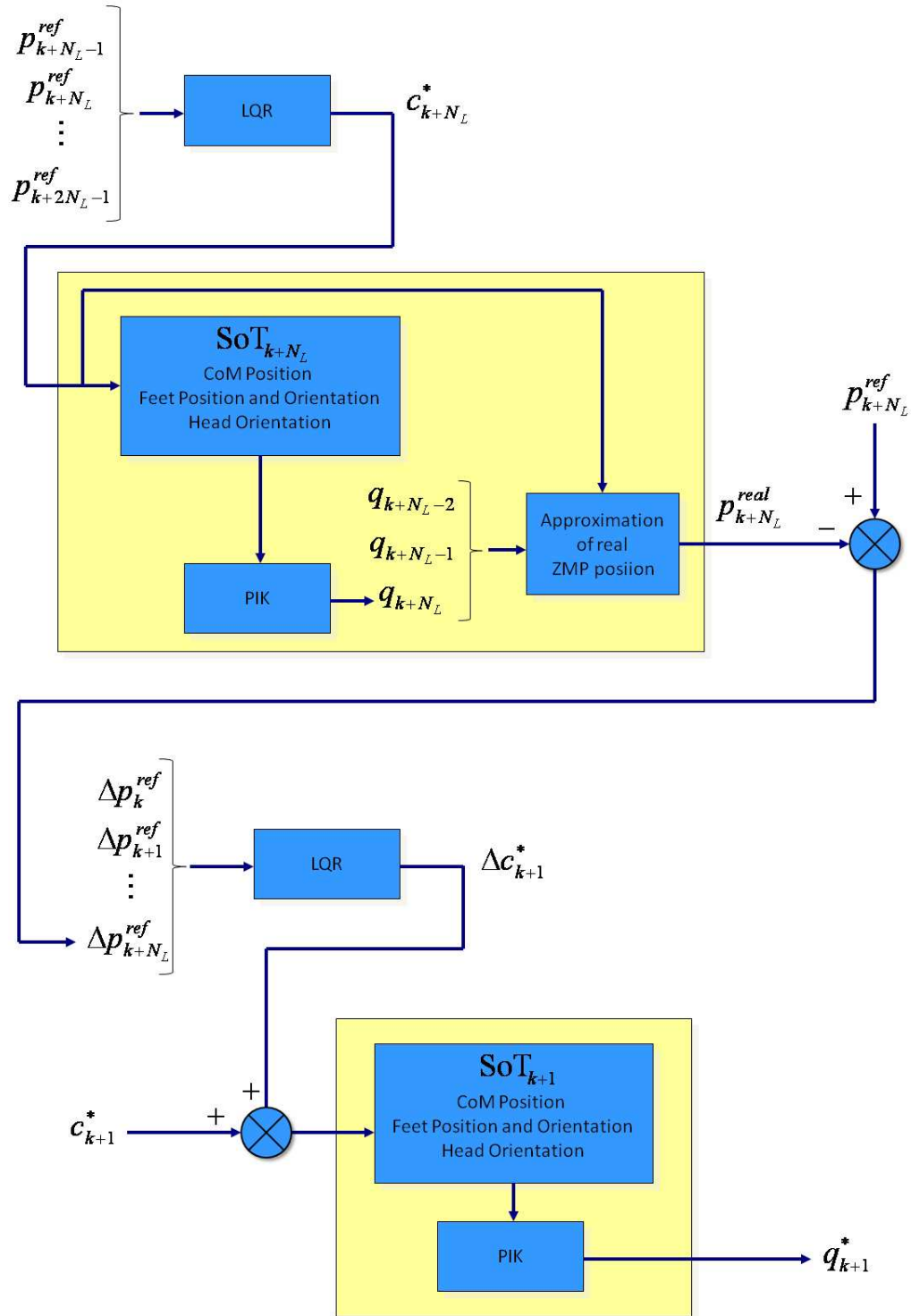


Figure 3.6 – Global architecture of the Walking Pattern Generator

3.4 Determination of the feet trajectories and head orientation

As established in the previous section, in addition to the CoM desired position, also the desired position and orientation of the robot feet, as well as the desired orientation of the head should be provided and thus construct a SoT which is to be submitted to a PIK process. This section introduces a series of methods to attain these purposes.

3.4.1 Feet trajectories

Not many scientific papers are available about the determination of the feet trajectories in walking pattern generation; a series of technical reports and unofficial documents can be found in the network. However, the specification of the feet trajectories can be done by using different techniques, which will be explained in this section.

During biped locomotion, there are two possible cases for the behavior of the robot feet, which are:

- Single support condition: The support foot is making contact with the ground while the floating foot is changing its position from one point to another.
- Double support condition: Both feet are making contact with the ground.

The first approach in the determination of the feet trajectories, regarding the x and y directions, is to set a linear interpolation for the floating foot describing the transition between $p_{x_{n-1}}^{ref}$ and $p_{x_{n+1}}^{ref}$, while the support foot position is maintained at $p_{x_n}^{ref}$, during

the single support interval $t_i < kT < t_f$, where $t_i = n(T_{ss} + T_{ds}) + \frac{T_{ds}}{2}$ and

$t_f = (n+1)(T_{ss} + T_{ds}) - \frac{T_{ds}}{2}$ are the initial and final time instants of this interval.

This means that the feet x and y positions during the single support interval are given by the following expressions:

$$\begin{aligned} foot_{sup}(k) &= p_n^{ref} \\ foot_{float}(k) &= \frac{p_{n+1}^{ref} - p_{n-1}^{ref}}{T_{ss}}(kT - t_m) + p_n^{ref} \end{aligned} \quad (3.15)$$

where $t_m = \frac{1}{2}(2n+1)(T_{ss} + T_{ds})$ is the central time instant of the single support interval.

During the double support interval, one foot position is kept at p_n^{ref} while the other foot stays at p_{n+1}^{ref} . An example of the behavior of the x -position of the left and right feet of a walking robot is illustrated in Figure 3.7.

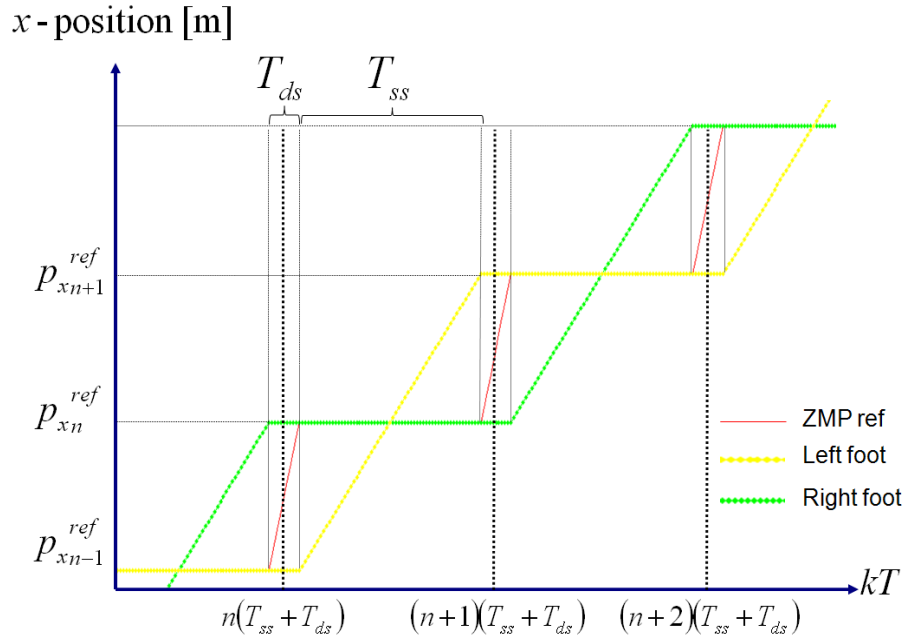


Figure 3.7 – Determination of the feet trajectories for x and y directions, by using linear interpolation.

The trajectory of the floating foot in the z -direction can be determined by assembling two segments of linear interpolation as follows:

$$foot_{float\ z}(k) = \begin{cases} \frac{z_{elev}}{T_{ss}}(kT - t_i) & t_i \leq kT < t_m \\ z_{elev} - \frac{z_{elev}}{T_{ss}}(kT - t_m) & t_m \leq kT \leq t_f \end{cases} \quad (3.16)$$

where z_{elev} is the maximum floating foot z elevation. These method exhibits such behavior as the one shown in Figure 3.8.

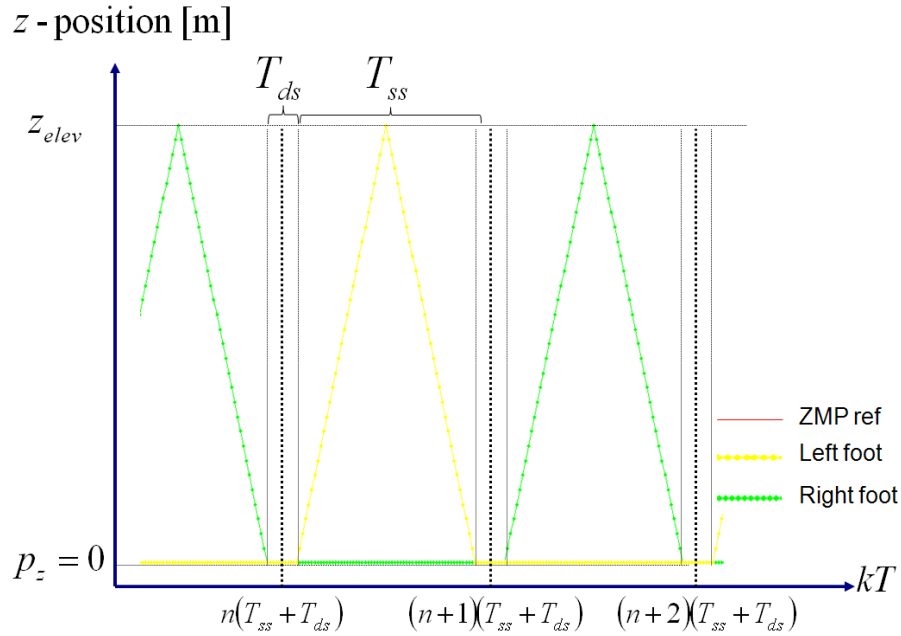


Figure 3.8 - Determination of the feet trajectories for z -direction, by using linear interpolation

Even though this first method of linear interpolation works adequately in the practice, it has the disadvantage that the resulting trajectories are continuous but not differentiable respect to time at certain instants. Therefore, a smoother and more effective approach is to use a third grade polynomial function to interpolate the behavior of the floating foot position; for x and y directions it is given by:

$$foot_{float}(k) = p_{n-1}^{ref} + \left(\frac{3(p_{n+1}^{ref} - p_{n-1}^{ref})}{T_{ss}^2} \right) (kT - t_i)^2 - \left(\frac{2(p_{n+1}^{ref} - p_{n-1}^{ref})}{T_{ss}^3} \right) (kT - t_i)^3 \quad (3.17)$$

and for z -direction an assembly of two similar interpolation polynomials is done in the following way:

$$foot_{float_z}(k) = \begin{cases} \left(\frac{3z_{elev}}{(T_{ss}/2)^2} \right) (kT - t_i)^2 - \left(\frac{2z_{elev}}{(T_{ss}/2)^3} \right) (kT - t_i)^3 & t_i \leq kT < t_m \\ z_{elev} - \left(\frac{3z_{elev}}{(T_{ss}/2)^2} \right) (kT - t_m)^2 + \left(\frac{2z_{elev}}{(T_{ss}/2)^3} \right) (kT - t_m)^3 & t_m \leq kT \leq t_f \end{cases} \quad (3.18)$$

These functions ensure that the velocity of the floating foot is equal to zero in all directions at the initial and final instants of the single support time interval and also that the floating foot reaches its maximum z elevation (z_{elev}) at instant t_m with velocity zero. An example of this technique is illustrated in Figure 3.9 and Figure 3.10.

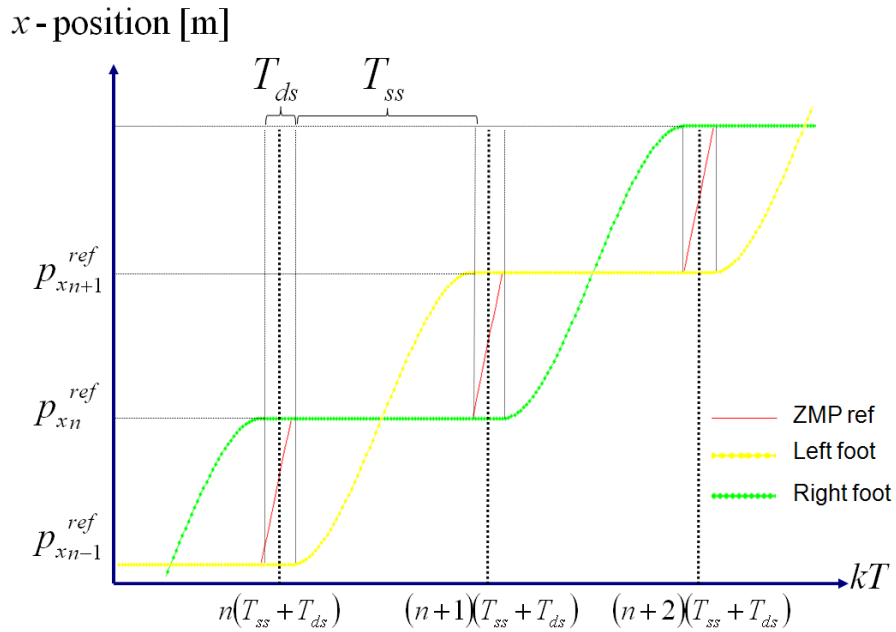


Figure 3.9 - Determination of the feet trajectories by using cubic interpolation

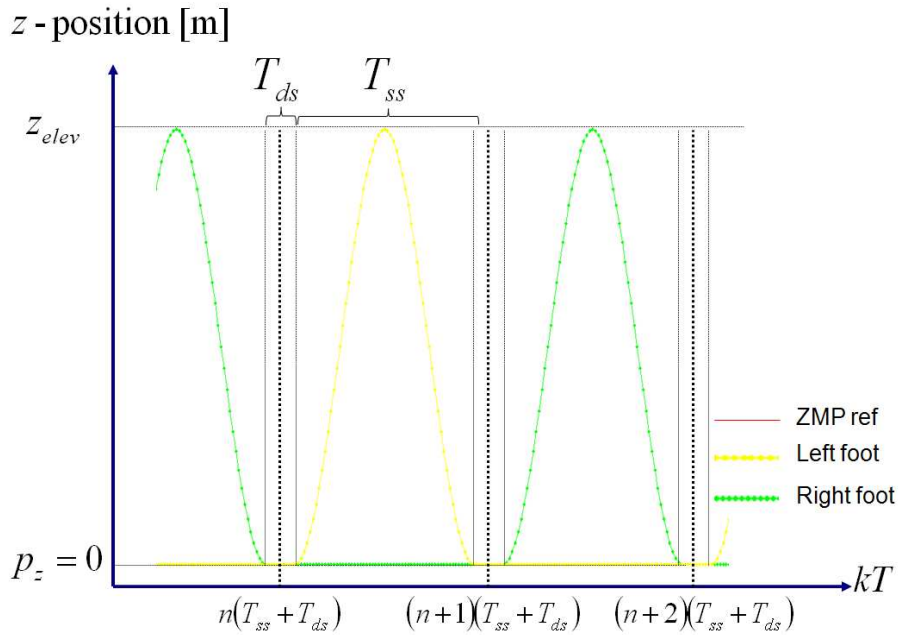


Figure 3.10 - Determination of the feet trajectories for z-direction, by using cubic interpolation

3.4.2 Head Orientation

The main objective of this work is to provide a Walking Pattern Generator for visually guided tasks; this means that the robot should be always keeping its head pointing towards a fixed landmark while it is walking from an initial to a final location.

To attain this goal, as stated in the previous section, the head orientation must also be included as a kinematic task into the SoT, This task is to be represented as an equality constraint whose error is given by:

$$e_{e_H}(q) = x_{H_{des}} - x_H(q) = \begin{pmatrix} p_{des} \\ \alpha_{des} \end{pmatrix} - \begin{pmatrix} p \\ \alpha \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ \psi_{des} \\ \theta_{des} \\ \phi \end{pmatrix} - \begin{pmatrix} x \\ y \\ z \\ \psi \\ \theta \\ \phi \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \psi_{des} - \psi \\ \theta_{des} - \theta \\ 0 \end{pmatrix} \quad (3.19)$$

The previous formulation means that the robot head is to be kept at the same position, while the Euler angles¹² ψ and θ of its orientation are to be modified to meet the desired values ψ_{des} and θ_{des} , respectively. Parameter ϕ of the head orientation does not need to be adjusted since the robot head can be pointing to the landmark regardless of the value of this angle.

The desired of the head orientation, can be obtained by calculating the difference between the landmark position vector L and the robot head position vector H , both expressed respect to the world reference frame, as shown in Figure 3.11.

¹² Consider Euler angles convention ZYX

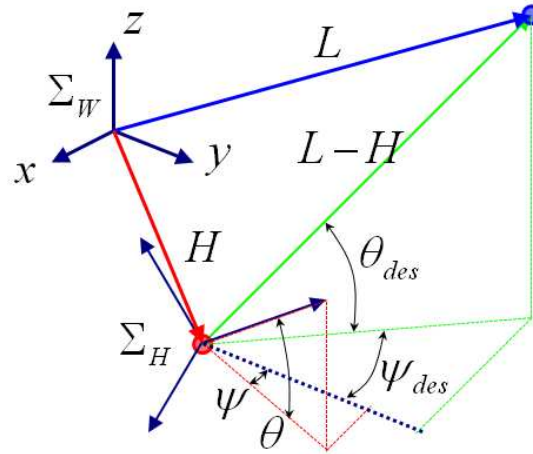


Figure 3.11 – Determination of the desired head orientation

Thus, the desired Euler angles can be determined as follows:

$$\begin{aligned}\psi_{des} &= \arctan\left(\frac{L_y - H_y}{L_x - H_x}\right) \\ \theta_{des} &= \arctan\left(\frac{L_z - H_z}{\sqrt{(L_x - H_x)^2 - (L_y - H_y)^2}}\right)\end{aligned}\quad (3.20)$$

where $L = \begin{pmatrix} L_x \\ L_y \\ L_z \end{pmatrix}$ and $H = \begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix}$.

Chapter 4

Simulations, experiments and results

As established in section 3.1, a proper validation of the Walking Pattern Generation designed in section 3.3 has been performed by using the humanoid robot NAO, whose general description is summarized in Appendix E.

This chapter presents the simulation and experimentation results of a series of examples of obtained walking patterns according to given paths, by using the global architecture illustrated in Figure 3.6, which characterizes the complete control scheme proposed in this thesis work.

The nature of the proposed paths represents in a very close way the human behavior when displacing from one location to another, while simultaneously maintaining visual contact with a fixed landmark point. In order to formulate these paths describing the locomotion of humanoid robots it was necessary to consider the non-holonomic constraint explained in section 2.4.2, which is typical of differential drive mobile robots, as proposed in [6].

The examples exposed in this chapter represent isolated characteristic behaviors, which can be assembled in order to generate a complete walking pattern for a real life application.

For each example a table indicating the relative footsteps-parameterized reference ZMP position is firstly provided, considering the notation illustrated in Figure 4.1:

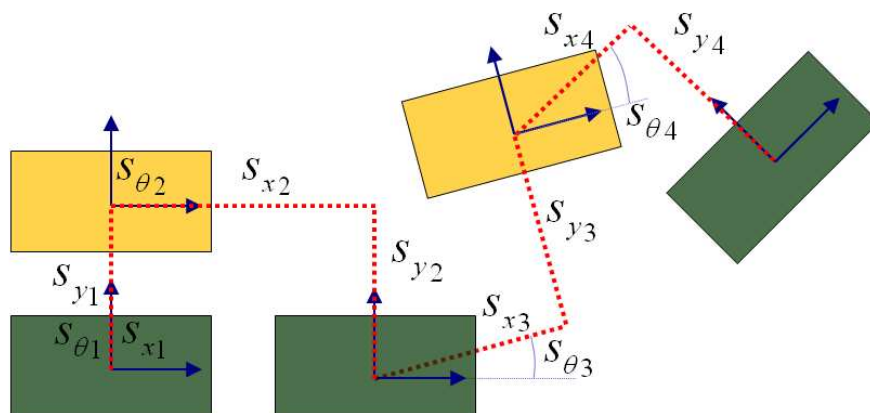








Figure 4.1 - Relative footstep parameterized reference ZMP position

The previous input information represents the foot positioning in a space $SE(2) = \mathbb{R}^2 \times SO(1)$, whose dimension is equal to 3, where the first two parameters define the foot position and the third parameter defines the foot orientation on the plane.

Subsequently, the results obtained from this input information after the execution of the Walking Pattern Generator algorithms are presented as absolute position trajectories for both x and y direction, as summarized in Table 4-1:

Table 4-1 Resulting trajectories provided for the developed simulations and experiments		
Trajectory	Symbol	Line color
Reference ZMP position (time sample parameterized)	ZMP_{ref}	
Linearized ZMP position (provided by the first stage of LQR)	ZMP_{pc}	
CoM position (provided by the first stage of LQR)	CoM_1	
Real ZMP position based on the robot movements (after first stage of PIK)	ZMP_{real1}	
Corrected CoM position (provided by the second stage of LQR)	CoM_2	
Real ZMP position based on the robot movements (after second stage of PIK)	ZMP_{real2}	

At the end of each example, a series of images extracted from the simulations in the software MATLAB are presented, as well as a set of photographs of the performed experiments on the NAO platform.

For the development of all presented examples in this section, the following input parameters for the preview controller were considered:

$$\begin{array}{llll}
 T = 0.01 \text{ s} & z_c = 0.27 \text{ m} & R = 1 \times 10^{-6} & z_c = 0.27 \text{ m} & N_L = 160 \\
 T_{ss} = 0.6 \text{ s} & p_z = 0 & Q_e = 1 & p_z = 0 & \\
 T_{ds} = 0.1 \text{ s} & & Q_x = 0 & z_{elev} = 0.15 z_c &
 \end{array}$$

and the following initial ZMP, CoM and feet absolute positions:

$$p_0 = \begin{pmatrix} 0 \\ 0 \\ p_z \end{pmatrix} \quad c_0 = \begin{pmatrix} 0 \\ 0.05 \\ z_c \end{pmatrix} \quad foot_{R0} = \begin{pmatrix} 0 \\ 0 \\ p_z \end{pmatrix} \quad foot_{L0} = \begin{pmatrix} 0 \\ 0.1 \\ p_z \end{pmatrix}$$

For the Prioritized Inverse Kinematics process, the stack of tasks was constructed by considering mixed systems of linear equalities and inequalities as shown in Table 4-2:

Table 4-2 Stack of Tasks in decreasing level or priority considered for the developed simulations and experiments

<i>P</i>	Equalities	Inequalities
1	CoM Position	Articular limits
2	Support foot position and orientation	Self-collision avoidance
3	Floating foot position and orientation	-
4	Head orientation	-

Moreover, the following parameters were considered for both PIK stages:

$cvTol = 0.01$	Convergence error tolerance
$cvdTol = 1 \times 10^{-6}$	Convergence error variation tolerance
$cvG = 0.5$	Convergence gain
$d_s = 0.03$	Obstacle avoidance security distance
$d_i = 0.04$	Obstacle avoidance influence distance
$Try_{max} = 500$	Maximum allowed iterations per sample
$\lambda_{ps} = 0.5$	Pseudoinverse matrix damping factor

4.1 Walking forward through a straight line path

The relative footstep parameterized reference ZMP position for this example was given to the Walking Pattern Generator as follows:

n	sx	sy	sQ
1	0	0.1125	0
2	0.05	-0.1125	0
3	0.05	0.1125	0
4	0.05	-0.1125	0
5	0.05	0.1125	0
6	0.05	-0.1125	0
7	0.05	0.1125	0
8	0.05	-0.1125	0
9	0.05	0.1125	0
10	0.05	-0.1125	0

with a fixed landmark located at position $L = (1 \ 1 \ 0.5)^T$ m .

The results provided by the Walking Pattern Generator are presented in Figure 4.2 and Figure 4.3, for x and y direction, respectively, considering the trajectories enlisted in Table 4-1. It is possible to identify that the ZMP deviation between the real and the referenced value exhibits a notable decreasing after the second regulation of preview control.

Figure 4.4 presents a set of simulation images of this example of walking pattern, considering the first four footsteps, and finally Figure 4.5 shows the corresponding real implementation on the humanoid robot NAO.

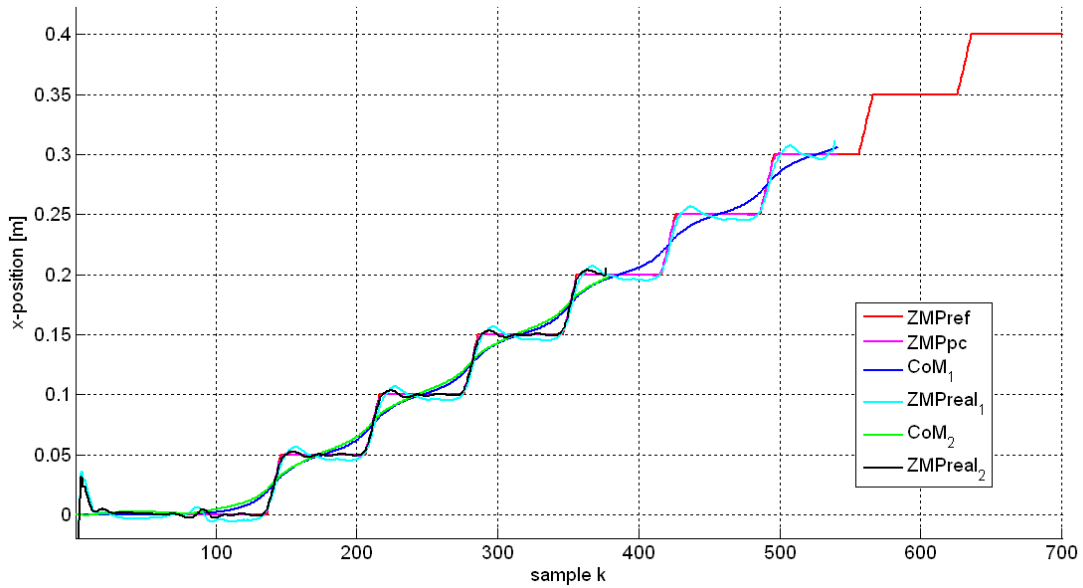


Figure 4.2 – Resulting trajectories in x -direction for a straight line forward walking pattern

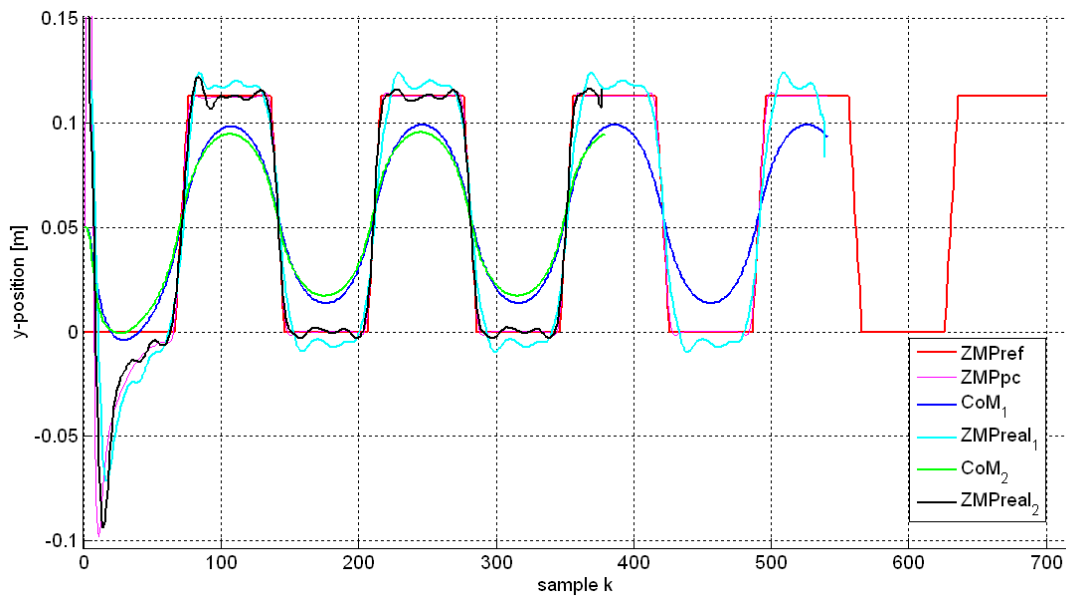


Figure 4.3 – Resulting trajectories in y -direction for a straight line forward walking pattern

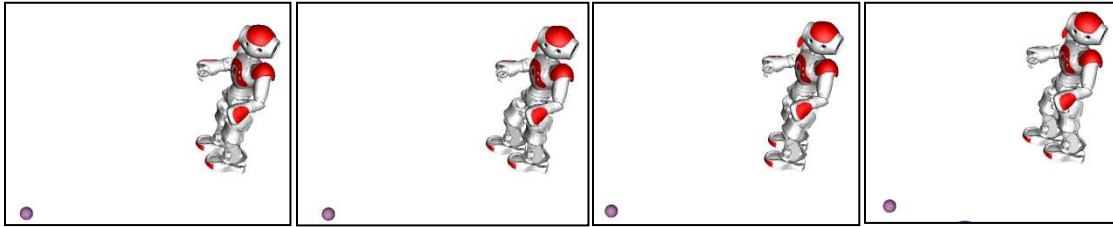


Figure 4.4 – Simulation images for a straight line forward walking pattern. The purple ball represents the landmark point.



Figure 4.5 – Images of the implementation of a straight line forward walking pattern

4.2 Walking backwards through a straight line path

This example is quite similar to the previous one, the only modifications are the negative signs in relative x ZMP reference positions s_{x_n} .

n	sx	sy	sQ
1	0	0.1125	0
2	-0.05	-0.1125	0
3	-0.05	0.1125	0
4	-0.05	-0.1125	0
5	-0.05	0.1125	0
6	-0.05	-0.1125	0
7	-0.05	0.1125	0
8	-0.05	-0.1125	0
9	-0.05	0.1125	0
10	-0.05	-0.1125	0

By regarding the ZPM and CoM graphs in Figure 4.6 and Figure 4.7, it is possible to recognize the change in the resulting trajectories for x direction, respect to the previous example, while the trajectories in y direction remain identical. Simulation and experiment images for the first four footsteps are shown in Figure 4.8 and Figure 4.9, respectively.

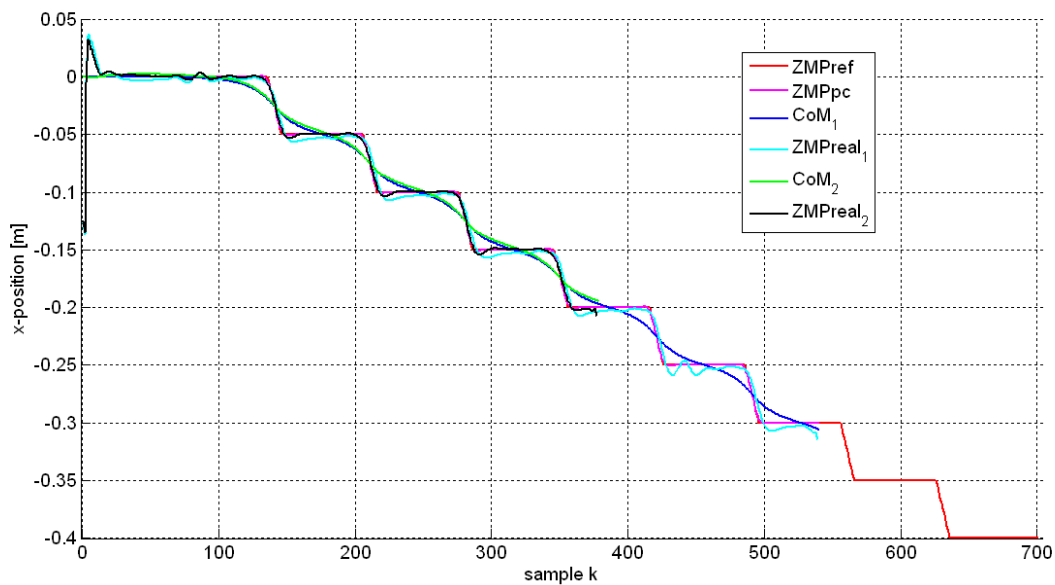


Figure 4.6 - Resulting trajectories in x -direction for a straight line backwards walking pattern

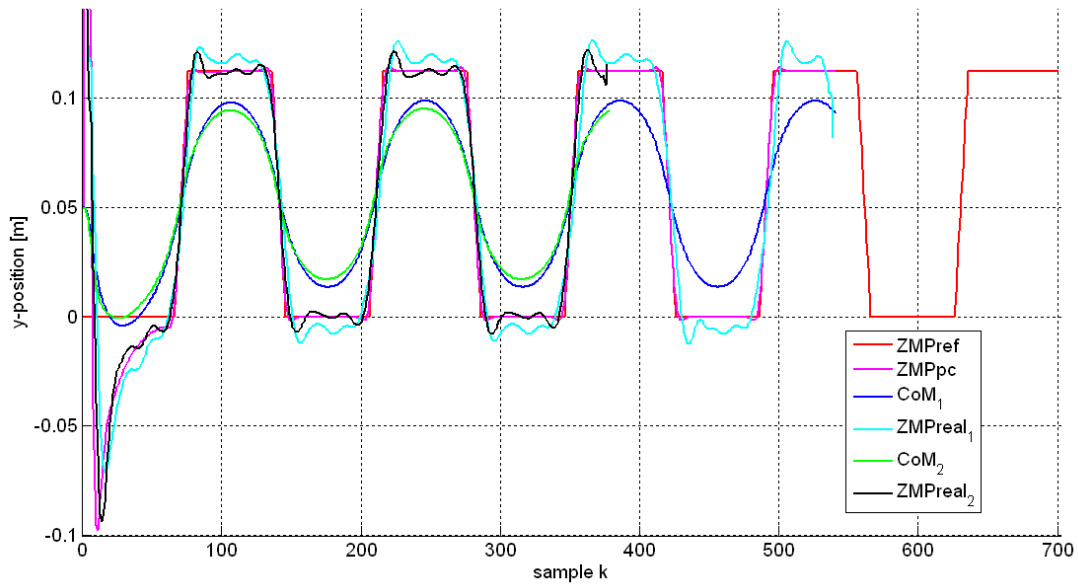


Figure 4.7 - Resulting trajectories in y-direction for a straight line backwards walking pattern

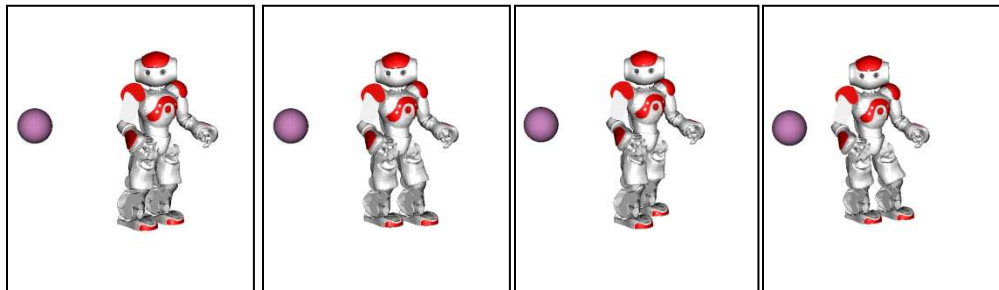


Figure 4.8 - Simulation images for a straight line backwards walking pattern. The purple ball represents the landmark point.



Figure 4.9 - Images of the implementation of a straight line backwards walking pattern

4.3 Walking forward through a curve line path

The example presented in this section describes the implementation of a directional walking pattern, whose concept allows the humanoid robot to displace to any location in $SE(2)$, by considering the non-holonomic constraint approached in section 2.4.2. This pattern considers, beside the relative foot positioning s_{x_n} and s_{y_n} , a relative foot angular position s_{θ_n} , which allows the robot to walk through curvilinear paths. For this example, the following input footstep data were utilized:

n	sx	sy	sQ
1	0.05	0.1125	15
2	0.05	-0.1125	10
3	0.05	0.1125	15
4	0.05	-0.1125	10
5	0.05	0.1125	0
6	0.05	-0.1125	-15
7	0.05	0.1125	-10
8	0.05	-0.1125	-15
9	0.05	0.1125	-10
10	0.05	-0.1125	0
11	0.05	0.1125	0
12	0.05	-0.1125	0

The data displayed in the table above corresponds to a pattern where the robot makes four steps forward while simultaneously turning a determined angle to its left; then one step forward with no angular change; subsequently four steps forward while turning to its right and finally three steps forward in straight line.

Figure 4.10 and Figure 4.11 show the results of the implementation of this example for x and y directions, respectively, as summarized in Table 4-1. By their part, Figure 4.12 and Figure 4.13 show the simulation and experimentation images illustrating the first eight footsteps of the developed example, which particularly describe the curvilinear nature of such kind of walking pattern.

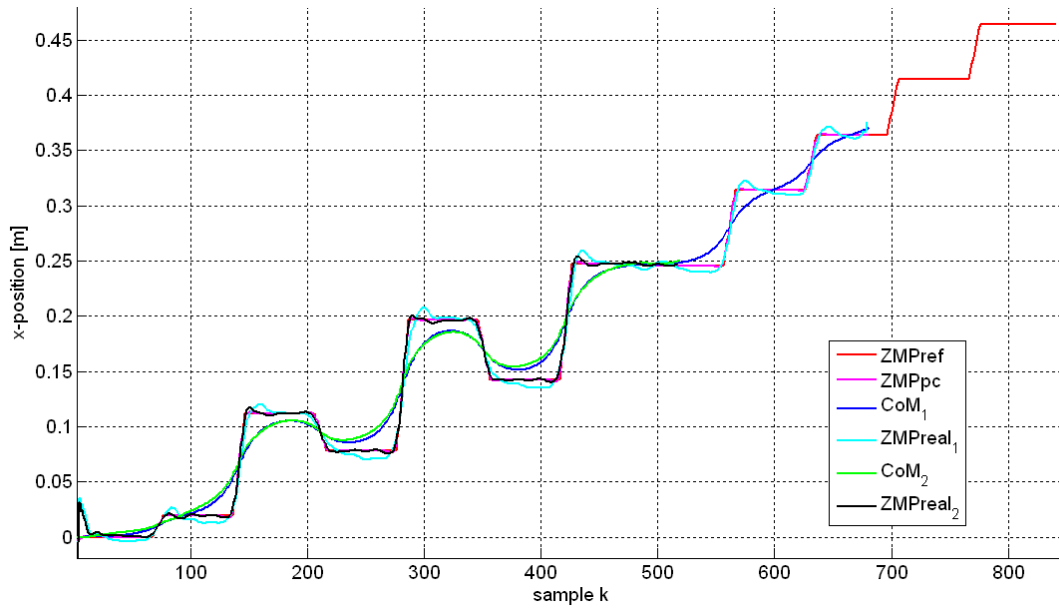


Figure 4.10 - Resulting trajectories in x-direction for a curve line forward walking pattern

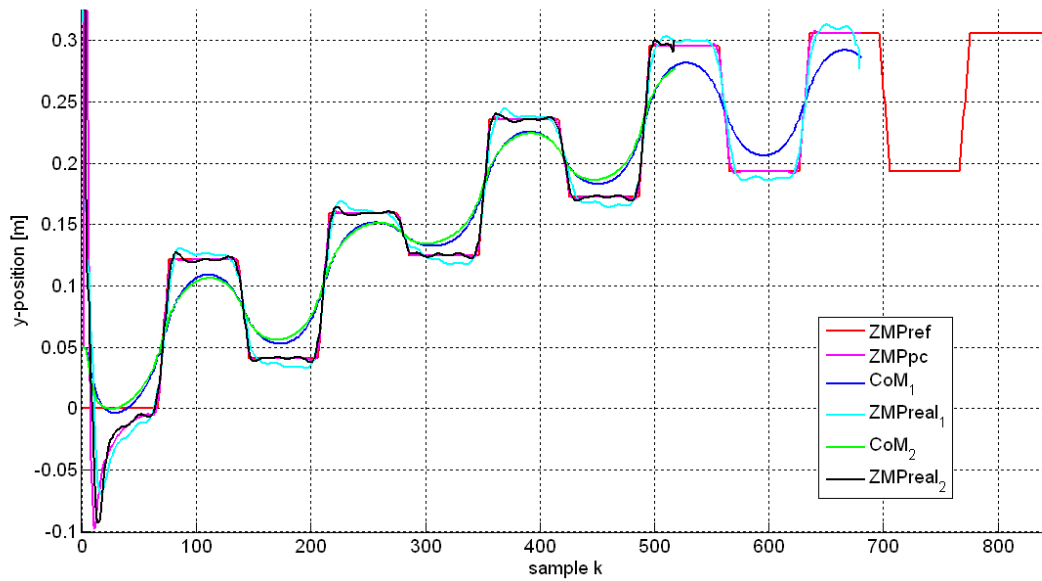


Figure 4.11 - Resulting trajectories in y-direction for a curve line forward walking pattern

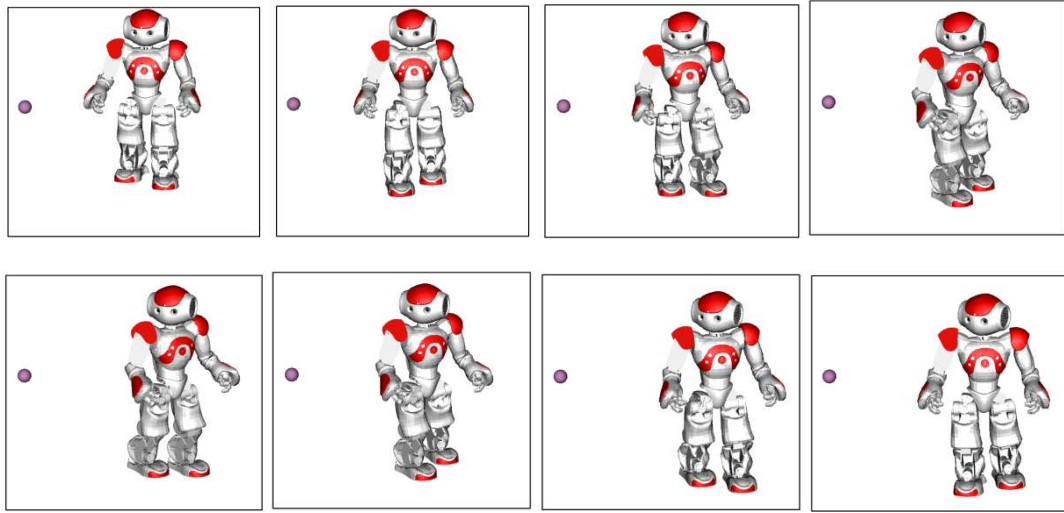


Figure 4.12 - Simulation images for a curve line forward walking pattern. The purple ball represents the landmark point.

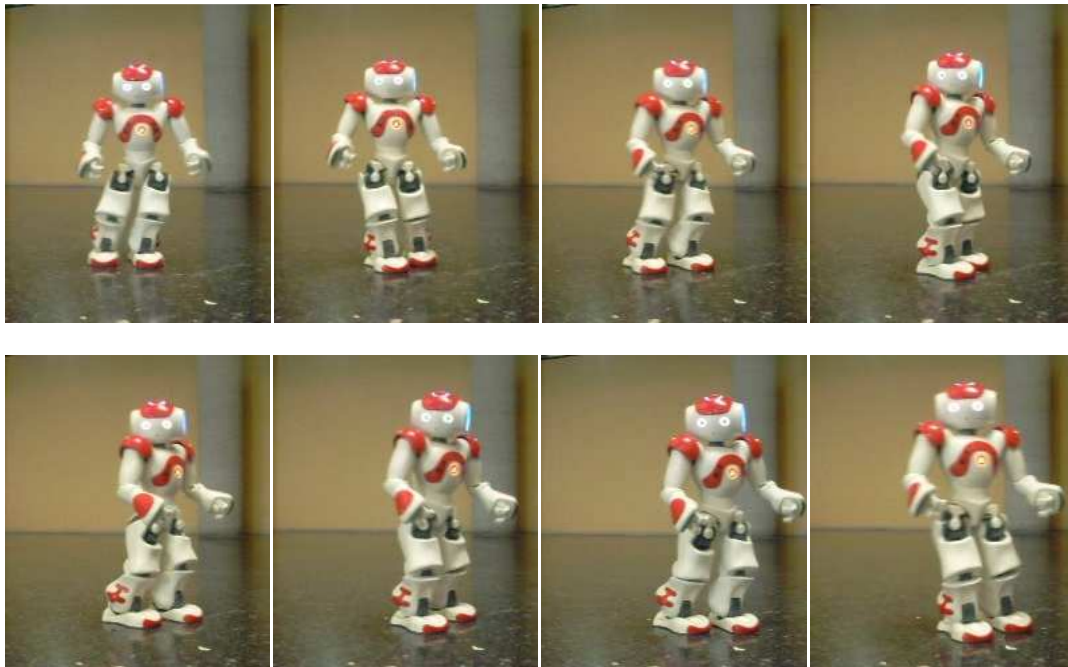


Figure 4.13 - Images of the implementation of a curve line forward walking pattern

4.4 Rotating around the vertical axis

Another important feature of mobile robots is the ability to turn around its own axis, with the purpose of changing its orientation without varying its current position. This kind of motion allows displacing through paths with corners at determined locations, while also keeping the non-holonomic constraint. Humanoid robots can perform this kind of self-axis rotation by locating their feet in such way as the one exposed in the present example:

n	sx	sy	sQ
1	0	0.1125	10
2	0	-0.1125	10
3	0	0.1125	10
4	0	-0.1125	10
5	0	0.1125	10
6	0	-0.1125	10
7	0	0.1125	10
8	0	-0.1125	10
9	0	0.1125	10
10	0	-0.1125	10

From the previous data it is possible to identify that there exist no changes in the feet relative x position, respect to the robot base frame (i.e., $s_{x_n} = 0 \forall n$), only the relative y position and the relative angular position exhibit a determine value; thus, the obtained pattern for this example is a counter-clockwise rotation around the vertical axis z .

Figure 4.14 and Figure 4.15 indicate the resulting ZMP and CoM graphs for this walking pattern in x and y directions, respectively. In Figure 4.16 and Figure 4.17 the corresponding simulation and implementation images are displayed, showing the first five steps of the pattern.

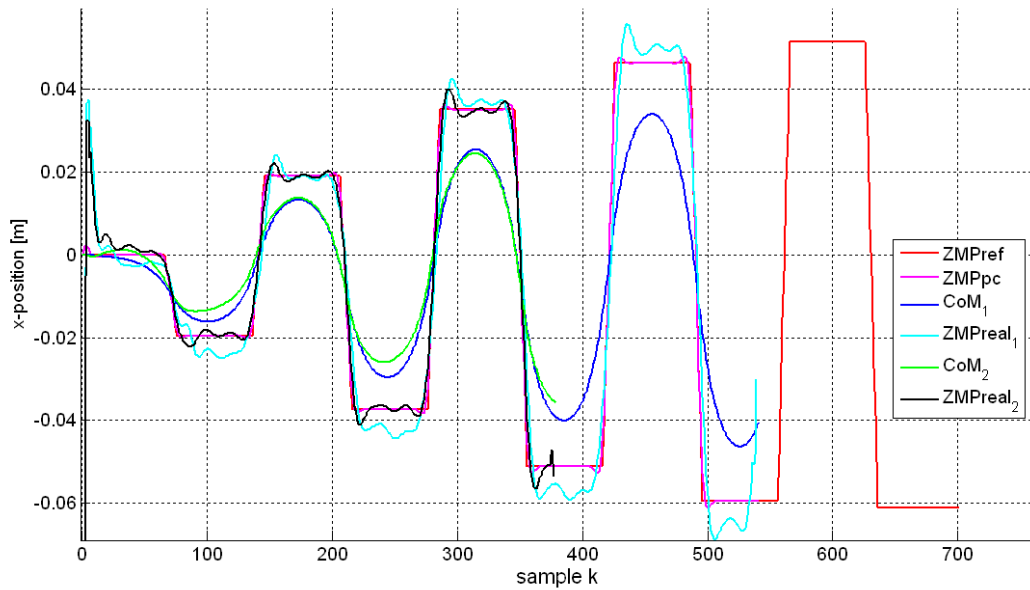


Figure 4.14 - Resulting trajectories in x -direction for a rotating around the vertical axis walking pattern

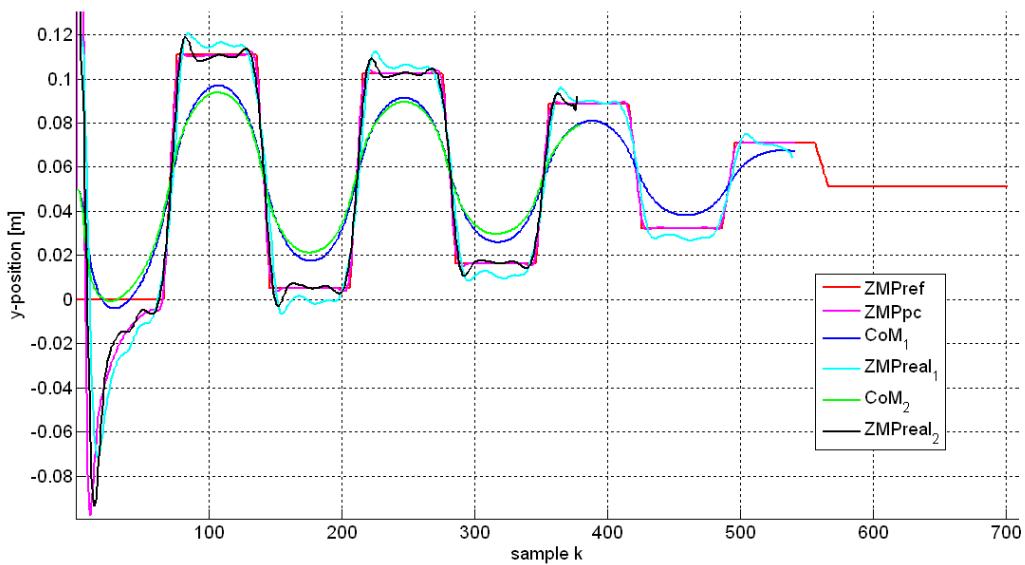


Figure 4.15 - Resulting trajectories in y -direction for a rotating around the vertical axis walking pattern

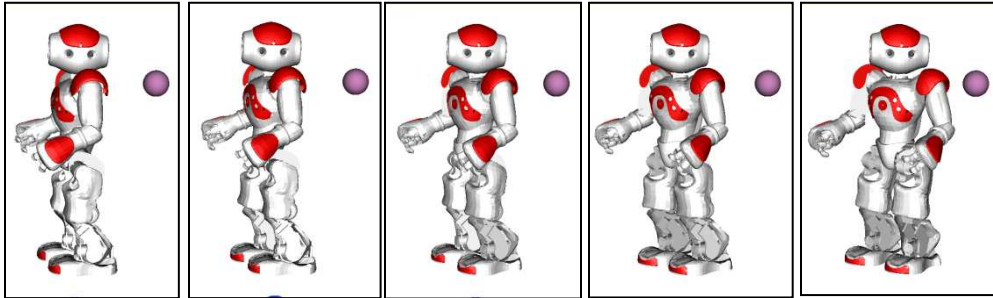


Figure 4.16 - Simulation images for a rotating around the vertical axis walking pattern.
The purple ball represents the landmark point.

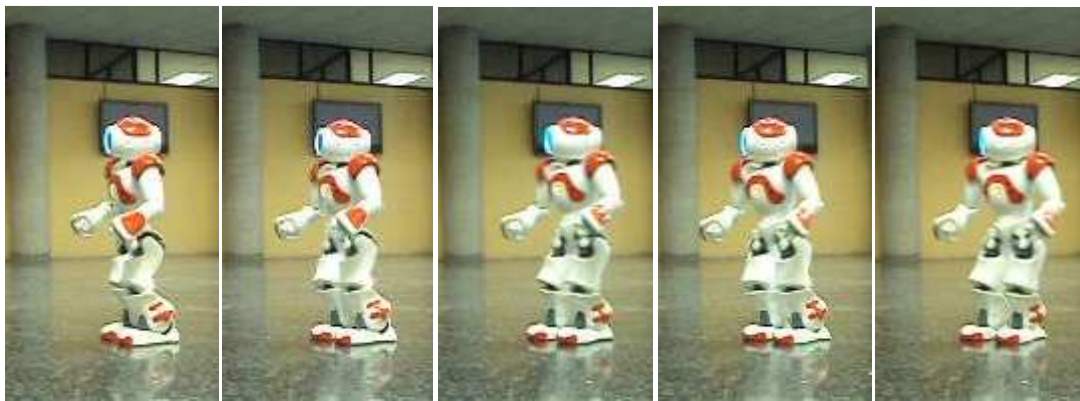


Figure 4.17 - Images of the implementation of a rotating around the vertical axis walking pattern

Chapter 5

Conclusions, Recommendations and Future Work

5.1 Conclusions

This master thesis work approached a wide series of geometric, kinematic and dynamic concepts, which were taken together to design a resourceful walking pattern generator for humanoid robots, suitable for visually-guided tasks.

In order to attain the objectives of the present project, a series of solutions to the most common problems present in the involved fields of study were proposed and the corresponding algorithms were also implemented in different programming engines.

Particularly, this work proposed, on one hand, a PIK scheme which takes profit of the robot redundancy to solve a stack of kinematic tasks by considering a decreasing order of hierarchy and demonstrating robustness to singularities. On the other hand, a powerful method was proposed to determine the optimal trajectory of the robot CoM based on the preview control of the reference ZMP trajectory.

Besides, there are currently no official works which specify the optimal priority order of the stack of kinematic tasks used to solve the PIK, nevertheless the best obtained results (i.e., faster and more accurate tasks convergence) in both virtual and real implementations correspond to the priority order proposed in Table 4-2.

The incorporation of a second stage of preview control and PIK reduced in a notable way the errors between the reference and real ZMP trajectories, due to the model simplifications. The disadvantage of utilizing a second stage of control is that the engine needs a wider future input reference window which causes that the final process time of the walking pattern generator becomes smaller.

After the algorithms implementation and their respective validation in both virtual and real platforms, it is possible to conclude that all the techniques used to design an efficient walking pattern generator can work successfully for practical applications.

5.2 Recommendations and Future Work

At this point, the developed walking pattern generator has been performed as an open-loop control system (i.e., the platform does not gather information about the current real environment conditions, in order to adjust the corresponding set points); moreover, the whole engine has been executed as an offline procedure which provides a sequence of articular configurations, from the given input reference, model parameters and set-up options. This offline generated output articular sequence was then loaded and executed into the simulation and experimentation platforms in order to test the system validity. The design of an online walking pattern generator which automatically takes real-time decisions according to incoming events is a potential future work research topic.

Another important area of opportunity, which is still being researched by the field experts, is the way to reduce the computation costs of the PIK algorithms in order to provide the output articular configuration in a faster time, while ensuring the task convergence. The computation time of the algorithms implemented for this project varied from the 10 milliseconds to the 50 milliseconds per iteration (according to the amount of priority levels and the type of algorithm used), which is not suitable yet for online applications.

Moreover, the present engine was designed to generate walking patterns for visually-guided tasks; this means that the robot head orientation was also considered as a kinematic task in the PIK procedure. The ability to control the head orientation inside the stack of tasks is also a wide and potential advantage for visual control applications, since this tool can provide a powerful close-loop online control scheme.

Furthermore, the present work assumed that the robot was walking on a flat plane, future research can also be done to provide walking patterns for irregular or multi-level environments, such as stairs, ramps or rocky ground.

Finally, special attention must be paid to the input reference of the ZMP position (i.e., the footsteps desired positions); although it is possible to propose this input information as a user-defined parameter; nonetheless, for applications with a higher amount of footsteps, there are currently some available engines for optimal path planning, such as the one proposed in [6], which consider the environmental constraints (landmarks visibility range, present obstacles) and the robot constraints (visibility range, non-holonomy and model dimensions).

Bibliography

- [1] **Kajita, S, et al.** *Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point*. Taipei, Taiwan : IEEE, 2003. IEEE International Conference on Robotics and Automation.
- [2] **Institute, Humanoid Robotics.** WABOT -WAseda roBOT-. [Online] Waseda University, Japan. [Cited: March 26, 2011.] http://www.humanoid.waseda.ac.jp/booklet/kato_2.html.
- [3] Honda Worldwide. *ASIMO - History*. [Online] 2011. [Cited: August 15, 2011.] <http://world.honda.com/ASIMO/history/history.html>.
- [4] **Robotics, Aldebaran.** *NAO User Guide Version 1.8.16*. Paris, France : s.n., 2006-2010.
- [5] Nitta Corporation. [Online] Nitta Corporation, 2011. [Cited: August 25, 2011.] <http://www.nitta.co.jp/english/>.
- [6] **Hayet, Jean-Bernard, Esteves, Claudia and Arechavaleta, Gustavo.** *Motion Planning for a Vigilant Humanoid Robot*. Paris, France : IEEE-RAS, 2009. IEEE-RAS International Conference on Humanoid Robots.
- [7] **Spong, M. W., Hutchinson, S. and Vidyasagar, M.** *Robot Modeling and Control*. s.l. : John Wiley & sons, 2006.
- [8] **Hootsmans, N. A. M. and Dubowski, S.** *Large motion control of mobile manipulators including vehicle suspension characteristics*. Sacramento, California : IEEE, 1991. IEEE International Conference on Robotics and Automation. pp. 63-71.
- [9] **Sugihara, T.** *Solvability-unconcerned inverse kinematics based on levenbergmarquardt method with robust damping*. Paris, France : IEEE-RAS, 2009. IEEE-RAS International Conference on Humanoid Robots. pp. 555-560.
- [10] **Nakamura, Y and Hanafusa, H.** *Inverse kinematic solutions with singularity robustness for robot manipulator control*. 1986, pp. 163–171.
- [11] **Baerlocher, Paolo and Boulic, Ronan.** *An Inverse Kinematic Architecture Enforcing an Arbitrary Number of Strict Priority Levels*. [ed.] N. Magnenat-Thalmann. s.l. : Springer, 2004, The Visual Computer: International Journal of Computer Graphics, Vol. 20, pp. 402-417.

-
- [12] **Kanoun, Oussama, et al.** *Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots*. Kobe, Japan : IEEE, 2009. IEEE International Conference on Robotics and Automation. pp. 724–729.
- [13] **The MathWorks, Inc.** Quadratic Programming - MATLAB. [Online] 1994-2011. <http://www.mathworks.com/help/optim/quadratic-programming.html>.
- [14] **Ferreau, Hans Joachim, et al.** *qpOASES User's Manual*. Leuven, Belgium : Katholieke Universiteit Leuven, 2008.
- [15] **Coleman, T.F. and Li, Y.** *A Reflective Newton Method for Minimizing a Quadratic Function Subject to Bounds on some Variables*. [ed.] Jorge Nocedal. s.l. : SIAM, 1996, SIAM Journal on Optimization, pp. 1040-1058.
- [16] **Escande, A., Mansard, N. and Wieber, P.-B.** *Fast resolution of hierarchized inverse kinematics with inequality constraints*. Anchorage, AK : IEEE, 2010. IEEE International Conference on Robotics and Automation. pp. 3733 – 3738.
- [17] **Kajita, S., et al.** *Introduction à la commande des robots humanoïdes*. [trans.] Sophie Sakka. France : Springer-Ferlag, 1999.
- [18] **Kajita, S., Matsumoto, O. and Saigo, M.** *Real-time 3D walking pattern generation for a biped robot with telescopic legs*. Seoul, Korea : IEEE, 2001. IEEE International Conference on Robotics and Automation. pp. 2299-2308.
- [19] **Kajita, S., et al.** *A Realtime Pattern Generator for Biped Walking*. Washington, DC. : IEEE, 2002. IEEE International Conference on Robotics and Automation. pp. 27-31.
- [20] **Kagami, S., et al.** *A Fast Generation Method of a Dynamically Stable Humanoid Robot Trajectory with Enhanced ZMP Constraint*. San Francisco, CA, USA : IEEE, 2000. Proceedings of the IEEE International Conference on Humanoid Robotics.
- [21] **Kajita, S., et al.** *The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation*. Maui, Hawaii, USA : s.n., 2001. IEEE/RSJ International Conference on Intelligent Robots and Systems.
- [22] **Takanishi, A., et al.** *Realization of Dynamic Biped Walking Stabilized by Trunk Motion on a Sagittally Uneven Surface*. Louis, MO, USA : IEEE, 1990. Proceedings of IEEE International Workshop on Intelligent Robots and Systems. pp. 323-330.
- [23] **Sheridan, T.B.** *Three Models of Preview Control*. s.l. : IEEE, 1966, IEEE Transaction on Human Factors in Electronics.
- [24] **Kayatama, T., et al.** *Design of an Optimal Controller for a Discrete Time System Subject to Previewable Demand*. 3, 1985, Int. J. Control, Vol. 41, pp. 677-699.

-
- [25] **Laub, Alan J.** *A Schur Method for Solving Algebraic Riccati Equations*. 6, s.l. : IEEE, 1979, IEEE Transactions on Automatic Control , Vol. 24, pp. 915-921.
- [26] **Robotics, Aldebaran.** Home - NAO's users. [Online] [Cited: August 12, 2011.] <http://www.aldebaran-robotics.com/>.
- [27] **Anderson, E., et al.** *LAPACK Users' Guide*. Third. Philadelphia, PA : Society for Industrial and Applied Mathematics, 1999. 0-89871-447-8.
- [28] **Faverjon, B. and Tournassoud, P.** *A local based approach for path planning of manipulators with a high number of degrees of freedom*. s.l. : IEEE, 1987. IEEE International Conference on Robotics and Automation. pp. 1152-1159.
- [29] **Sugihara, T., Nakamura, Y. and Inoue, H.** *Realtime Humanoid Motion Generation through ZMP manipulation based on Inverted Pendulum Control*. Washington DC : IEEE, 2002. Proceedings of the IEEE International Conference on Robotics and Automation. pp. 1404-1409.
- [30] **Gray, Andrew.** *A Treatise on Gyrostatics and Rotational Motion*. London : Macmillan, 1918. ISBN 978-1-4212-5592-7.
- [31] **Belongie, Serge.** Rodrigues' Rotation Formula. [Online] MathWorld - A Wolfram Web Resource. [Cited: August 22, 2011.] <http://mathworld.wolfram.com/RodriguesRotationFormula.html>.
- [32] **Rockfeller, R. T.** *Convex Analysis*. s.l. : Princeton University Press, 1970.
- [33] **Gouaillier, David, Collette, Cyrille and Kilner, Chris.** *Omni-directional Closed-loop Walk for NAO*. Nashville, TN, USA : IEEE-RAS, 2010. IEEE-RAS International Conference on Humanoid Robots.
- [34] **Goswami, A.** *Posturel stability of biped robots and the foot-rotation indicator(fri) point*. 1999, International Journal of Robotics Research, pp. 523-533.

Appendix A: Convention of Euler Angles

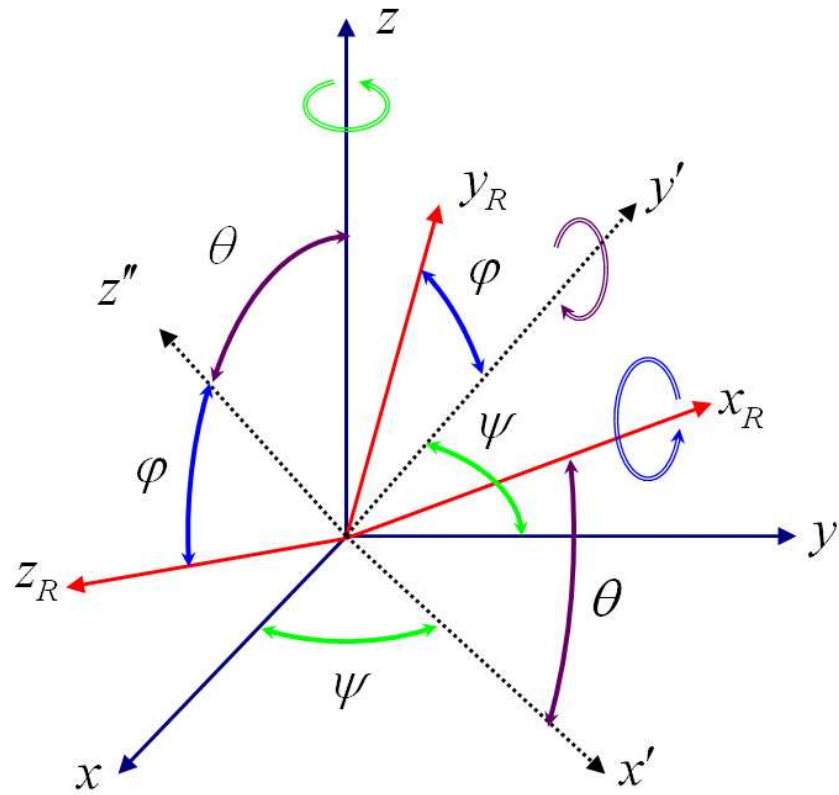
The Euler angles [30] are three angles introduced by Leonhard Euler to describe the orientation of a rigid body. In order to describe an orientation in the tridimensional Euclidean space, three parameters are required. These parameters can be given by using several notations, where the Euler angles is one of them.

The Euler angles represent three composed rotations that move a reference frame Σ_W , denoted by (x, y, z) , to a rotated frame Σ_R , denoted by (x_R, y_R, z_R) ; this means that any orientation can be achieved by composing three elemental rotations around a single axis, whose values are the Euler angles, which are commonly denoted by φ , θ and ψ . As a result of this, any rotation matrix can be decomposed as a product of three elemental rotation matrices.

There are different conventions of Euler angles, which are divided into two groups; one of them is called proper Euler angles and the other is called Tait-Bryan angles. Any kind of convention considers three rotations around a single axis. Proper Euler angles are equivalent to three combined rotations repeating exactly one axis (e.g., ZXZ, XYX, or YZY). Tait-Bryan angles are equivalent to three composed rotations in different axes (e.g., ZYX, YZX, or ZXY).

Furthermore, for any Euler angles convention, intrinsic or extrinsic rotation equivalence can also be used, giving two different meanings for the same convention name; therefore this parameter should also be specified. Intrinsic rotations are performed around the mobile frame axes while extrinsic rotations are considered around the reference frame axes. The first type is the most commonly used for Euler angles conventions, since the value of the three intrinsic rotations represent directly the Euler angles values.

The figure bellow illustrates the particular Euler angles convention that was used for this thesis work, which is the ZYX with intrinsic composition. It means that firstly, a rotation around the z -axis is done with a value of ψ , taking x to x' and y to y' ; subsequently, a rotation around the y' -axis is done with θ , taking x' to x_R and z to z'' ; and finally, a rotation around the x_R -axis is done with φ , taking y' to y_R and z'' to z_R .



Euler angles convention ZYX with intrinsic composition

Appendix B: Equation of Rodrigues

In order to obtain a rotation matrix R from a rotation velocity vector $\hat{\omega}$, it is necessary to consider the corresponding rotation velocity matrix \dot{R} , which is given by:

$$\dot{R} = \hat{\omega}R$$

The previous equation is a differential equation, respect to the matrix variable R . By considering $R(0) = I$ and a constant rotation velocity vector, then the solution of this differential equation is given by the exponential matrix $e^{\hat{\omega}t}$, as follows:

$$R(t) = e^{\hat{\omega}t} = I + \hat{\omega}t + \frac{(\hat{\omega}t)^2}{2!} + \frac{(\hat{\omega}t)^3}{3!} + \dots$$

By considering the relative rotation axis vector a , introduced in section 2.5.4, the rotation velocity vector can be expressed as $\omega = a|\omega|$, and due to the characteristic $\hat{a}^3 = -\hat{a}$ (\hat{a} is anti-symmetric and $|a|=1$), then any \hat{a}^n of high power can be expressed as \hat{a}^2 . Moreover, by using Taylor series of sine and cosine functions, the following equation is obtained:

$$e^{\hat{\omega}t} = I + \hat{a} \sin(\omega t) + \hat{a}^2 (1 - \cos(\omega t))$$

which is called **Equation of Rodrigues** [31], and provides the rotation matrix corresponding to a constant rotation velocity vector. Furthermore, in the previous equation it is possible to consider that $\omega t = \theta$, where θ is the rotation angle; then it can be rewritten as:

$$e^{\hat{a}\theta} = I + \hat{a} \sin(\theta) + \hat{a}^2 (1 - \cos(\theta))$$

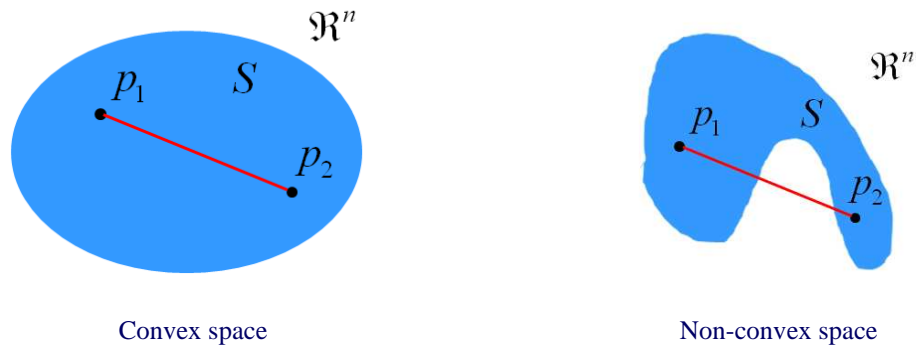
which is a commonly used relation in kinematics calculations.

Appendix C: Definition of Convex Hull

A subspace S that belongs to \mathfrak{R}^n is considered to be **convex** [32] if the condition

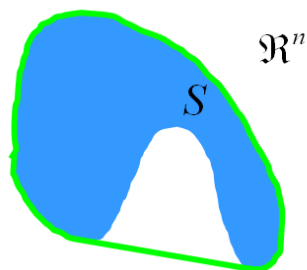
$$\alpha p_1 + (1 - \alpha)p_2 \in S$$

is satisfied for every $p_1, p_2 \in S$, with a scalar α such that $0 \leq \alpha \leq 1$. It is possible to identify that, when $n=2$, the previous condition implies that if a segment constituted from the connection of two arbitrary points p_1 to p_2 of S remains also inside of S , then S is considered to be a convex space, as illustrated in the following figure:



Definition of convex space

Moreover, being S a subspace of \mathfrak{R}^n , the smallest convex space that contains the subspace S is denominated **convex hull**, denoted as $\text{co}S$. From a more intuitive perspective, the smallest convex space that includes the subspace S is generated by placing an elastic string around S , as shown in the next figure:



Generation of a convex hull

In the particular case where S is constituted by a finite set of points, the corresponding convex hull is a convex polyhedron.

Appendix D: Solution of the discrete-time algebraic Riccati equation by using RSF representation

A discrete-time algebraic Riccati equation is defined in the following way:

$$F^T X F - X - F^T X G_1 (G_2 + G_1^T X G_1)^{-1} G_1^T X F + H = 0$$

where $F, H, X \in \mathfrak{R}^{n \times n}$, $G_1 \in \mathfrak{R}^{n \times m}$, $G_2 \in \mathfrak{R}^{m \times m}$; moreover, $H = H^T > 0, G_2 = G_2^T > 0$ and also $m < n$.

According to the method and assumptions established in [25], with the purpose of finding the solution X of the discrete-time algebraic Riccati equation, firstly by setting $G = G_1 G_2^{-1} G_1^T$ the following symplectic matrix should be considered:

$$Z = \begin{pmatrix} F + G F^{-T} H & -G F^{-T} \\ -F^{-T} H & F^{-T} \end{pmatrix}$$

Since Z has no eigenvalues on the unit circle, it is possible to find an orthogonal transformation $U \in \mathfrak{R}^{2n \times 2n}$, which puts Z in *Real Schur Form (RSF)* as follows:

$$U^T Z U = S = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix}$$

Where S is a quasi-upper triangular matrix and the blocks $S_{ij} \in \mathfrak{R}^{n \times n}$. Moreover, it is possible to arrange that the spectrum of S_{11} lies inside the unit circle, while the spectrum of S_{22} lies outside the unit circle.

Furthermore, U can be partitioned conformably as

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$$

From the previous considerations, U_{11} is invertible and $X = U_{21} U_{11}^{-1}$ is the solution of the discrete-time algebraic Riccati equation.

Appendix E: General description of the NAO platform

NAO is a small (56cm) and light (4.8Kg) fully actuated biped robot provided by the French company Aldebaran Robotics [4]. Since the beginning of the company in July 2005, a major goal has been the development of robust walk for the robot.

It is constituted by 25 DoF (5 in each leg, 1 in the pelvis, 2 in the head, 5 in each arm and 2 actuated hands). It is equipped with two cameras, an inertial measuring unit, sonar sensors in its chest, and force-sensitive resistors under its feet, as illustrated in the following figure:

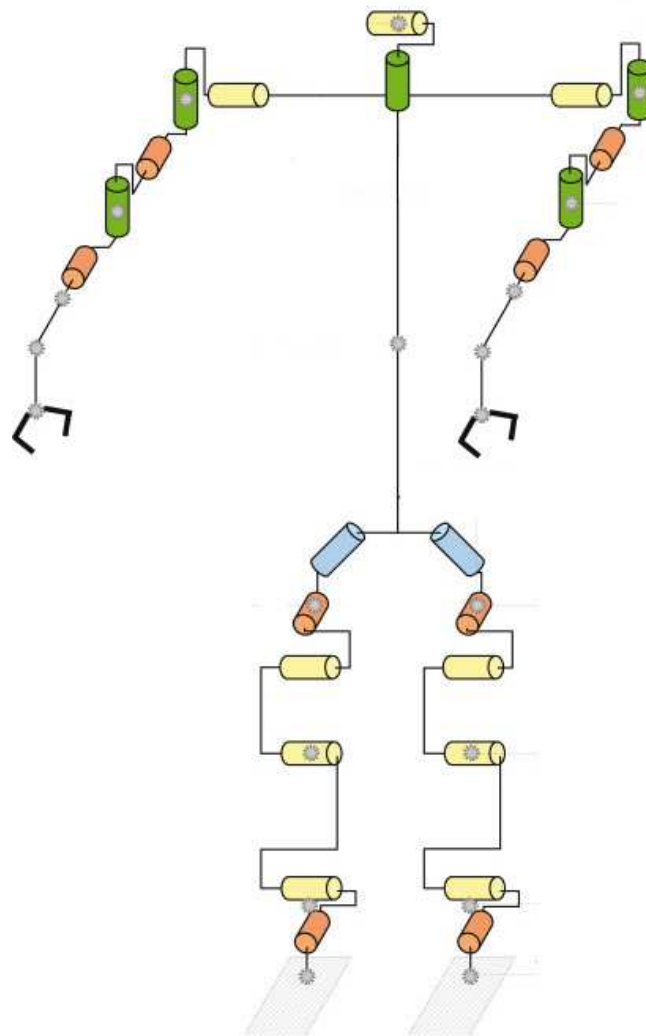


Sensors and Actuator of the humanoid robot NAO

NAO is different to all other biped platforms because it's an affordable robot (approximately 12000 Euros) which is fully programmable at high level or low level with Aldebaran Robotics Software Development Kit.

After 5 years of mecatronic design and improvements in robustness of the robot (7 prototypes) and multiple prototypes of humanoid dynamic walk algorithm, an omni-directional walk engine [33] robust against small obstacles is now available for all the NAO units (more than 700) in the world.

A depiction of the kinematic structure of the NAO platform is presented in the following figure, where the cylinders represent the model articulations and the lines represent the model bodies.



Kinematic model of the NAO robot

Furthermore, the following tables present the geometric and dynamic parameters of the bodies that constitute the NAO humanoid robot, according to the modeling techniques presented in section 2.5.2.

Body parameter		BASE	RIGHT LEG					
Self ID		1	2	3	4	5	6	7
Mother ID		0	1	2	3	4	5	6
Mass		1.02628	0.07244	0.1353	0.39798	0.29706	0.13892	0.16304
axis vector (respect to mother frame)	x	0	0	1	0	0	0	1
	y	0	0.7071068	0	1	1	1	0
	z	0	0.7071068	0	0	0	0	0
relative position vector (respect to mother frame)	x	0	0	0	0	0	0	0
	y	0	-0.05	0	0	0	0	0
	z	0	-0.085	0	0	-0.1	-0.10274	0
position of CoM (respect to local frame)	x	-0.0048	-0.00717	-0.01649	0.00131	0.00471	0.00142	0.02489
	y	6.00E-05	0.01187	-0.00029	-0.00201	-0.0021	-0.00028	-0.0033
	z	0.0423	0.02705	-0.00475	-0.05386	-0.04891	0.00638	-0.03208
end Point (respect to local frame)	x	0	0	0	0	0	0	0
	y	0	0	0	0	0	0	0
	z	0	0	0	-0.1	-0.10274	0	-0.04511
Inertial Matrix (respect to local frame) [Kg.m ²]	xx	0.0049674	9.065E-05	2.791E-05	0.0016656	0.0011803	3.883E-05	0.0002676
	xy	-1.25E-05	-4.89E-06	4.982E-08	4.325E-07	3.054E-07	-4.94E-08	-5.93E-06
	xz	-0.000164	-1.22E-05	5.31E-06	-8.25E-05	-4.02E-05	-2.21E-06	-0.000137
	yx	1.247E-05	-4.89E-06	4.982E-08	4.325E-07	3.054E-07	-4.94E-08	-5.93E-06
	yy	0.0047914	0.0001067	0.0001015	0.001625	0.0011293	7.212E-05	0.0006431
	yz	2.478E-05	2.784E-05	4.9E-10	3.174E-05	3.574E-05	6.74E-09	1.87E-05
	zx	-0.000195	-1.22E-05	5.31E-06	-8.25E-05	-4.02E-05	-2.21E-06	-0.000137
	zy	2.225E-05	2.784E-05	4.9E-10	3.174E-05	3.574E-05	6.74E-09	1.87E-05
zz	0.0015671	6.743E-05	9.209E-05	0.0003049	0.0001935	5.352E-05	0.0005265	
joint upper limit (in degrees)		0	-90	-45	-100	0	-75	-25
joint lower limit (in degrees)		0	90	25	25	130	45	45

Body parameter		LEFT LEG					
Self ID		8	9	10	11	12	13
Mother ID		1	8	9	10	11	12
Mass		0.07244	0.1353	0.39798	0.29706	0.13892	0.16304
axis vector (respect to mother frame)	x	0	1	0	0	0	1
	y	0.7071068	0	1	1	1	0
	z	-0.707107	0	0	0	0	0
relative position vector (respect to mother frame)	x	0	0	0	0	0	0
	y	0.05	0	0	0	0	0
	z	-0.085	0	0	-0.1	-0.10274	0
position of CoM (respect to local frame)	x	-0.00717	-0.01649	0.00131	0.00471	0.00142	0.02489
	y	-0.01187	0.00029	0.00201	0.0021	0.00028	0.0033
	z	0.02705	-0.00475	-0.05386	-0.04891	0.00638	-0.03208
end Point (respect to local frame)	x	0	0	0	0	0	0
	y	0	0	0	0	0	0
	z	0	0	-0.1	-0.10274	0	-0.04511
Inertial Matrix (respect to local frame) [Kg.m2]	xx	9.065E-05	2.791E-05	0.0016656	0.0011803	3.883E-05	0.0002676
	xy	-4.89E-06	4.982E-08	4.325E-07	3.054E-07	-4.94E-08	-5.93E-06
	xz	-1.22E-05	5.31E-06	-8.25E-05	-4.02E-05	-2.21E-06	-0.000137
	yx	-4.89E-06	4.982E-08	4.325E-07	3.054E-07	-4.94E-08	-5.93E-06
	yy	0.0001067	0.0001015	0.001625	0.0011293	7.212E-05	0.0006431
	yz	2.784E-05	4.9E-10	3.174E-05	3.574E-05	6.74E-09	1.87E-05
	zx	-1.22E-05	5.31E-06	-8.25E-05	-4.02E-05	-2.21E-06	-0.000137
	zy	2.784E-05	4.9E-10	3.174E-05	3.574E-05	6.74E-09	1.87E-05
zz	6.743E-05	9.209E-05	0.0003049	0.0001935	5.352E-05	0.0005265	
joint upper limit (in degrees)		-90	-45	-100	0	-75	-25
joint lower limit (in degrees)		90	25	25	130	45	45

Body parameter		HEAD	
Self ID		14	15
Mother ID		1	14
Mass		0.05959	0.47671
axis vector (respect to mother frame)	x	0	0
	y	0	1
	z	1	0
relative position vector (respect to mother frame)	x	0	0
	y	0	0
	z	0.1265	0
position of CoM (respect to local frame)	x	-0.00003	0.00383
	y	0.00018	-0.00093
	z	-0.02573	0.05156
end Point (respect to local frame)	x	0	0
	y	0	0
	z	0	0.04
Inertial Matrix (respect to local frame) [Kg.m2]	xx	6.223E-05	0.002089
	xy	4.2E-10	5.491E-06
	xz	7.448E-08	0.0001134
	yx	4.2E-10	5.491E-06
	yy	6.324E-05	0.0019322
	yz	1.042E-07	-2.8E-05
	zx	7.448E-08	0.0001134
	zy	1.042E-07	-2.8E-05
	zz	5.495E-06	0.0008226
joint upper limit (in degrees)		-120	-45
joint lower limit (in degrees)		120	45

Body parameter		RIGHT ARM			
Self ID		16	17	18	19
Mother ID		1	16	17	18
Mass		0.06984	0.12166	0.05959	0.112
axis vector (respect to mother frame)	x	0	0	1	0
	y	1	0	0	0
	z	0	1	0	1
relative position vector (respect to mother frame)	x	0	0	0.09	0
	y	-0.098	0	0	0
	z	0.1	0	0	0
position of CoM (respect to local frame)	x	-0.00178	0.002067	-0.02573	0.06992
	y	0.02507	-3.88E-03	0.00001	-0.00096
	z	0.00019	0.00362	-0.0002	-0.00114
end Point (respect to local frame)	x	0	0.09	0	0.12855
	y	0	0	0	0
	z	0	0	0	-0.0159
Inertial Matrix (respect to local frame) [Kg.m ²]	xx	7.102E-05	5.932E-05	5.495E-06	5.395E-05
	xy	-2.02E-06	-1.94E-06	-2.24E-08	5.382E-06
	xz	-1.71E-08	1.565E-05	-6.77E-08	3.561E-06
	yx	-2.02E-06	-1.94E-06	-2.24E-08	-2.16E-06
	yy	1.405E-05	0.0002022	6.225E-05	0.0002308
	yz	-4.41E-08	-3.69E-06	5.59E-09	-4.72E-08
	zx	-1.71E-08	1.565E-05	-6.77E-08	-5.47E-06
	zy	-4.41E-08	-3.69E-06	5.59E-09	-4.72E-08
	zz	7.314E-05	0.0001842	6.323E-05	0.0002279
joint upper limit (in degrees)		-120	-95	-120	0
joint lower limit (in degrees)		120	0	120	90

Body parameter		LEFT ARM			
Self ID		20	21	22	23
Mother ID		1	20	21	22
Mass		0.06984	0.12166	0.05959	0.112
axis vector (respect to mother frame)	x	0	0	1	0
	y	1	0	0	0
	z	0	1	0	1
relative position vector (respect to mother frame)	x	0	0	0.09	0
	y	0.098	0	0	0
	z	0.1	0	0	0
position of CoM (respect to local frame)	x	-0.00178	2.07E-03	-0.02573	0.06992
	y	-0.02507	0.00388	-0.00001	-0.00096
	z	0.00019	0.00362	-0.0002	-0.00114
end Point (respect to local frame)	x	0	0.09	0	0.12855
	y	0	0	0	0
	z	0	0	0	-0.0159
Inertial Matrix (respect to local frame) [Kg.m ²]	xx	7.102E-05	5.932E-05	5.495E-06	5.395E-05
	xy	-2.02E-06	-1.94E-06	-2.24E-08	5.382E-06
	xz	-1.71E-08	1.565E-05	-6.77E-08	3.561E-06
	yx	-2.02E-06	-1.94E-06	-2.24E-08	-2.16E-06
	yy	1.405E-05	0.0002022	6.225E-05	0.0002308
	yz	-4.41E-08	-3.69E-06	5.59E-09	-4.72E-08
	zx	-1.71E-08	1.565E-05	-6.77E-08	-5.47E-06
	zy	-4.41E-08	-3.69E-06	5.59E-09	-4.72E-08
	zz	7.314E-05	0.0001842	6.323E-05	0.0002279
joint upper limit (in degrees)		-120	0	-120	-90
joint lower limit (in degrees)		120	95	120	0