ÉCOLE CENTRALE DE NANTES

MASTER CORO-IMARO

"CONTROL AND ROBOTICS"

2020 / 2021

Master Thesis

Presented by

Selvakumar Hastham Sathiya Satchi Sadanandam

On 23 August 2021

Realtime human motion imitation by humanoid robot with balance constraint

Evaluators:	Dr. Olivier Kermognant	Associate Professor (LS2N, ECN)
	Dr. Christine Chevallereau	Director of Research (IRCCyN, CNRS)
Supervisor(s):	Dr. Sophie Sakka	Assistant Professor (LS2N, ECN)

Laboratory: Laboratoire des Sciences du Numérique de Nantes LS2N

Abstract

Humanoid robots are made as mirror to the humans. Consequently, the humanoid motion is also expected to be real as human and it is natural to use human motion as an input to generate humanoid motion like a child learning from recogniting an action and imitating. This process in robots is called motion imitation and there are several challenges posed due to kinematics and dynamics of the robot over the past decades. Although the work on the kinematic challenges is actively improving and notably better than dynamics, it allows the robot only to move and imitate slow actions. For the fast paced motions, *momentum* gets build up and needs dynamics to be taken into account. Due to the differences in redundancy between humanoid robots and humans, real-time imitation in humanoid robots while keeping balance and support changes is still an challenging problem that need to be addressed.

On this research, motion retargetting or the motion imitation based on dynamic balance and support with dynamic filtering control is implemented on humanoid robot *NAO v5 H25*, from Aldebaran Robotics. The imitation will be carried out online using marker-based motion capture suit from *Xsens*, specifically *Xsens MVN Analyze* system.

The captured motion from the inertial suit will be preprocessed for its representation in operational space and will be scaled to the robot's dimensions. From this scaled motion, for this thesis, a set of actions are taken and will be addressed for the balance problem in *NAO* robot. The joint, CoM and the ZMP data of the human actor will be mapped to the robot using the scaling function directly. To ensure the stability and to keep up the speed with the human actor, an additional dynamic filter and multi-inverted pendulum based posture control will be implemented.

Acknowledgements

I would first like to thank my thesis advisor Dr Sophie Sakka for her motivation and opportunity for the thesis topic. Her experience, knowledge and continuous advicess have allowed me to become a better researcher and have ultimately made me to progress in the research problem. I am specifically grateful to the members in *Robots! Association* for their support and providing a very friendly environment during my thesis period.

I would also like to thank Dr Olivier Kermorgant and Dr Christine Chevallereau for accepting to be members of the jury, for their interest in my work and for their feedback on the thesis.

Finally, my deepest thanks go out to my parents for their constant support, advice, love and help not only during the development of this thesis, but throughout all of my life. Thanks for always being there when I most needed it, in spite of the distance; thanks for all the encouragements at all times; thanks for all the beliefs I was always taught. And, overall, thanks God for the life and for always giving me the strength to keep forward despite all the difficulties.

Notations

x_i	Pose State i
X	Pose State Vector
q	Joint State Vector
$\dot{q}$	Joint Velocity Vector
$\ddot{q}$	Joint Acceleration Vector
A_H	Parameter A of Human Actor
A_R	Parameter A of Humanoid Robot
r_i	Position vector of body i
m_i	Mass of body i
M	Total mass of the robot
ξ	Generalized velocity vector
τ	Generalized torque
$M(\ddot{q})$	Generalized Inertial Matrix of Robot
$c(q, \dot{q})$	Vector of Coriolis, Centrifugal and Gravity
I_i	Inertia matrix of body i
$\mathbf{g}$	Gravity vector
P_{CoM} or X_{CoM}	Position of CoM of the robot
P_{CoM_i}	Position of CoM of body i
P_{ZMP} or X_{ZMP}	Position of ZMP
$\mathcal{F}_i$	Frame representation of body i respecting M-DH
$\mathcal{R}_i$	Local coordinate system of body i
κ	Normalization matrix or vectors in constraints
Q	Weighing matrix in QP

Abbreviations

CoM	Centre of Mass
DoF	Degrees of Freedom
ZMP	Zero Moment Point
OSID	Operational Space Inverse Dynamics
QP	Quadratic Programming
HQP	Hierrachical Quadratic Programming
eHQP	Equality-constrained Hierrachical Quadratic Programming
iHQP	Inequality-constrianed Hierrachical Quadratic Programming
M-DIP	Multi- Double Inverse Pendulum Model
SoT	Stack of Tasks
MVNX	Xsens-specific XML format
UDP	User Datagram Protocol

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The interest of humans in building, controlling and researching complex machines that look and behave like humans is not recent; it started very early in humanity. There has always been numerous models of human-like machines over the centuries crafted by the engineers, mathematicians and craftsman that prove the existence of human-alike machines. After the development of Unimate and Shakey, the robotics has been widely boosted. Indeed, the first Industrial revolution of robotics only considered arm manipulators and wheeled platforms in a structured and defined environments. The interest towards human-like robots or humanoid robots has always grown instead of being only in science fiction. The recent researches and development has proved the possiblity of building humanoid robots or simply humanoids, becoming reality, although not powerful or autonomy as desired.

In order to develop more autonomous interactions in humanoid robots, one of the major and main objective is that it should be able to imitate human motions precisely. This would be a major milestone to be achieved in the field of humanoid robots. Scientists and researchers, over the past decades are thriving to make the humanoid robot movements as close as human motions. Recently, the ability of the humanoid robots to teleoperate increases rapidly. To imitate the motion perfectly, the robot should be able to understand the motion; hence, the importance of motion capture systems has widely increased in robotics especially in humanoid robots.

Nowadays, a variety of technologies exist that allow for high accurate capturing of human motions with high frequency. By imitating captured motion, humanoid robots can be teleoperated and also learn new skills resembling human actions. However, there is a catch because the direct imitation of captured movements is impossible due to the differences in the

degree of freedoms and the weight distributions between humans and humanoids. Depending on the complexity of the motion, the challenge of motion generation increases due to various humanoid's constraints including the constraints in stability and the extended period of imitation. This chapter presents the brief introduction on the humanoid robots, their development and behaviour towards human-like motions.

1.1 Service Robotics

According to *International Federation of Robotics (IFR)*, a service robot is defined as a machine that performs useful tasks for humans or equipment excluding industrial automation applications. These robots are mainly dedicated to accompany humans with possibly reduced capabilities and to assist them in dull, dangerous or repetitive tasks. Moreover, *IFR* divides the service robots into two possible categories.

- *Personal Service Robots* - These robots are used for non-commercial and social tasks. The robots are usually trained to operate with non-trained sociologist rather engineers or roboticists.

- *Professional Service Robots* - These robots are used for a commercial task and are usually operate with well-trained operators. The robots tend to work in a professional or well-defined environments are built based on the stack of specialized tasks.

The service robots that are mentioned previously tend to be application-depending and are able to designed to satisfy the specific needs. One of the main aim of the service robots is to co-exist with humans alongside and to help them with any tasks that humans may need assist. However, typical human environments are completely different to the environments that robots experience currently. The robot environments are well-defined with a set of deterministic or stochastic variables that allows the robots for better estimations. Though the robots are able to estimate the environment to an extent, it's abilities are nowhere near for human capabilities. Another important problem is to build a robot that can perform multiple tasks and hold multiple applications. Hence, for a robot to exist among humans and to interact naturally with people, one robot must overcome all these difficulties to some degree of autonomy. Therefore, one of the major challenges in service robotis it to build a general robot that can succeed in all

the areas where human beings can. Equivalently, there is a challenge to build robots that also act and behave as humans.

1.2 Humanoid robots

Humanoid robots are expected to exist and work in a close relationship with human beings in the everyday world and to serve the needs of physically handicapped people. These robots must be able to cope with the wide variety of tasks and objects encountered in dynamic unstructured environments. Imagining an humanoid robot collaborates with humans to execute some daily tasks, learn actions from humans and even improve it's ability to teleoperate [7]. When a humanoid robot works in collaboration with human, the interaction through gestures and cooperation is essential.

Besides the satisfaction of human curiosity and imagination, the integration of humanoid robots in our daily lives makes sense for a variety of practical reasons. Robots resembling us would make human-robot interaction more natural and thus more intuitive and pleasurable. Moreover, if robots are to assist people in daily chores, they have to fit the human environment, which is suitable for *human morphology* [8]. Performing tasks with two hands, handling tools, climbing stairs, reaching shelves - to name only a few tasks - require our assistants to have a similar morphology to ours, so robots can adapt to human lives, instead of us having to do the opposite. Furthermore, research in humanoid robots directly contributes to the field of prosthesis and exoskeletons.

The challenges in creating such machines are numerous. Human bodies are energy efficient machines with extremely elaborated mechanics and incredible cognitive and perceptual abilities. Thus, the creation of humanoid robots requires advances in more areas than one, from the improvement of sensors and processing of information, to efficient control techniques and suitable mechanical structures with constraints in shape, size and weight to improve the resemblance to human beings.

The idea of building machines which look and move like humans has been explored by philosophers and mathematicians since antiquity. Nowadays, the concepts of such machines are a part of research in robotics. Humanoid robots can be thought of as mechanical, actuated devices that can perform human-like manipulation, including locomotion as their main skill for

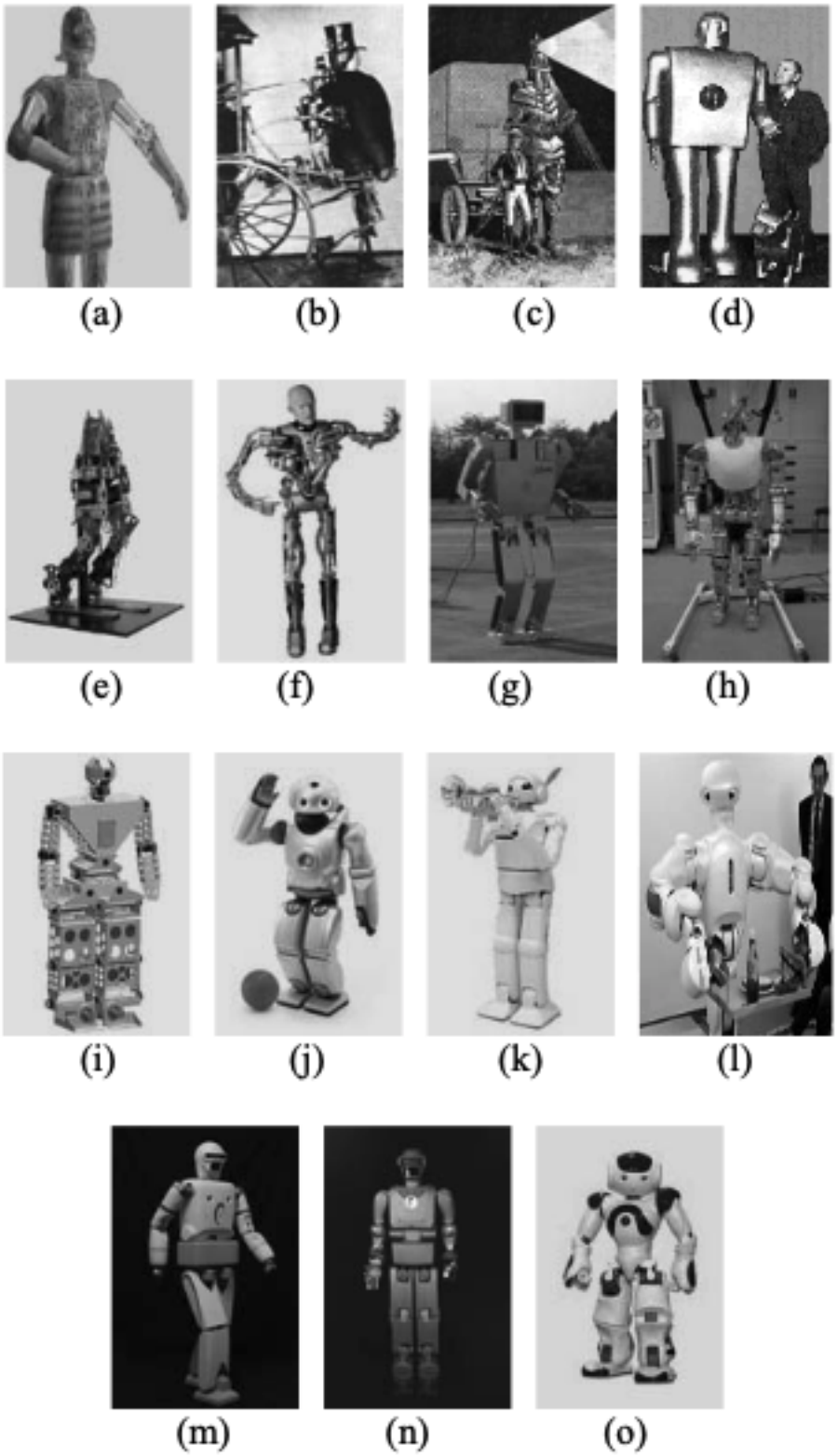

Figure 1.1: Some Bipedal Android robots over decades [9] (a) First humanoid robot by Leanardo Da vinci (b) Steam Mam in 1865 (c) Electric man in 1885 (d) ELEKTRO in 1938 (e) BIPER - 4 in 1984 (f) Tron-XM in 1997 (g) H6 Humanoid robot in 2000 (h) Robot JACK in 2000 (i) GuRoo in 2002 (j) QRIO, Sony in 2003 (k) Partnar Robot, Toyoto in 2004 (l) TwentyOne in 2007 (m) REEM-A in 2007 (n) REEM-B in 2008 (o) NAO in 2008.

displacement. Well before the first modern humanoid robot, one of the biggest steps towards this objective was achieved in 1956 with the first commercial robot manipulator, *Unimate, from Unimation*. The automotive industry was the first to benefit from these kinds of manipulator robots. Recently, the development of humanoid robots for education, research and services has proved the work in multiple ways.

Current goals of research in humanoid robots include industrial and social applications in day-to-day life. A study was conducted by Tanie, [10] and is briefed below

- maintenance tasks of industrial plants,
- security service for home and offices,
- human care, teleoperation of construction machines.
- cooperative work

Since many studies have explored this aspect of robot motion and how to make robots more *human-like* and *human-aware*. Human Robot Interaction (HRI) is now a challenging research field and studies on the efficacy of humanoid robots in human environments are further proceeded.

1.3 Measuring Human Movement

Kinesiology is defined as the scientific study of human movement. To access human motion, Kinesiology involves principles and methods from *biomechanics, anatomy, physiology and motor learning*. Its range of application includes health promotion, rehabilitation, ergonomics, health and safety in industry, disability management, among others. The measurement of human movement is one of the tools that is central in this research field. In 19th century, various devices were built to produce the moving pictures, among those exists the most advanced technique named *chronophotography*. This device allowed to study fast paced human motions by recording and reproducing the captured motion [11]. Another major contribution in motion study is the study on the path of center of mass during human displacement [12].

The first experimental studies of human gait [12], i.e. determining physical quantities like inertial properties, were conducted by Christian W. Braune (1831-1892) and Otto Fischer(1861-

Figure 1.2: Computation of human locomotion using differential equations by Weber Brothers [12]. The coordination between arms and legs was observed clearly.

1917). They considered the human body as rigid bodies in form of dynamic links in series. The work of Nicholas Bernstein (1896-1966) in Moscow introduced the 3D analysis based on cameras. The methods for measuring human movement continued improving with the advent of new technologies like electronics and magnetic devices, up until today's motion capture systems based on reflective markers, magnetic or inertial devices. Recently, there has been a huge development in the motion capture systems which evolved the motion study to newer dimensions (discussed in the later chapter).

1.4 Challenges in Motion Imitation

Given the resemblance between humanoid robots and human beings, it is natural to look for inspiration in human movements in order to generate motion for the humanoid. The most straightforward way is to have the robot observe what the human does and reproduce that behaviour, i.e. perform imitation. After all, even human beings themselves are able to acquire skills by imitation and learning.

If humanoid robots are to interact with human beings, it is imperative that their gestures are human-like since much of human communication is non-verbal. Programming each aspect of the motion detail by detail in order to make it human-like is time-consuming and not fit to handle the immense variety and complexity of human behaviours. Thus, imitation comes as a more natural and intuitive alternative to classical methods, since more information can be transmitted directly.

Human motion cannot be directly transferred to the robot without choosing beforehand which affects the ocean or what is transferring. The most advanced human robots cannot move

Figure 1.3: A performance based on motion imitation done by HRP-2 humanoid robot

completely like a human being. The robots are limited by differences to human counterparts such as the number of degrees of freedom, link lengths, motor torques, etc. Robots which tried to reproduce the whole human body inevitably have never taken dynamic differences from the human body they are trying to represent.

While performing imitations, the physical differences have to be taken into account when mapping the register home movements to the robot morphology. This was addressed by Pollard et al. [5], who limited the captured human motion to a range achievable by the robot by locally scaling angles and velocities in order to preserve as much as possible local variations in the motion imitated by the Sarcos robot.

1.5 Problem Statement

Recently, almost every humanoid robots are able to walk and balance in flat indoor environments and there are robots proved walking on rugged terrains and uneven planes are possible and achievable. A lot of effort is being done to make them more autonomous by incorporating the perception, planning and action loop. One of the ultimate objective of the humanoid robot, as mentioned before is to create the humanoid motion more human-precise. In this sense, robots require real-time imitation processed be higher reactivity compensating the unpredictable na-

ture of human motion.

In humans, imitation is an advanced behavior whereby an individual observes and replicates the action of another human arguably with more accuracy and precision. However in humanoid robots, these kind of motion imitations are possible up to kinematic level through perception; the robot can also be able to imitate the action posture dynamically using its predefined configurations and controllers to an extent. But for a complete imitation at dynamic level, humanoid robots are still struggling to approximately copy the dynamic parameters applied during the action. Presently, to copy the action at dynamic level, feedback data from human during motion or action is mandatory. The motion data from human action is transferred using *Xsens MVN Analyze*. The main objective of the thesis is to define realtime dynamic balance constraint during motion imitation and validate it experimentally using an affordable humanoid robot, *NAO* from *Aldebaran Robotics*. The balancing constraint of the robot is controlled using an Hierachical Quadratic Programming (HQP) and validated on both simulation and real robot.

1.6 Thesis Organisation

The chapters of the thesis are organized as follows. Chapter 2 introduces the generalities of humanoid robotics and presents the state of the art in the control and balance of this type of robots. The existing works on motion retargetting involving dynamics are also discussed. Chapter 3 presents the mathematical requirements for the motion imitation problem. The approaches for CoM and ZMP retargetting are also explained in this chapter. Posture Control during motion imitation using multiple double inverted pendulums are proposed. Chapter 4 presents the essential concepts of hierrachical quadratic programming (HQP) and presents the tuned approach for retargetting problem. Chapter 5 describes the environmental setup of the proposed system. Information on Xsens and NAOqi are detailed in this chapter. Chapter 6 details the implementation of HQP and their corresponding results. The final chapter 7 concludes the overall work, discussion and the future scope of the thesis.

Chapter 2

State of the Art

The research and interest in humanoid robotics has greatly increased over the last few years, but inspite of the current abundance of the humanoid robots, their utility is still very limited. One of the most important trials, *DARPA Robotics Challenge (DRC)* in 2013 listed a pack of capabilities and robustness that humanoid robots lacked when performing different tasks. Each task of this challenge should last only up to 30 minutes. These tasks will take less than a minute for a human to complete which explains the powerlessness of the robots. This chapter briefly explains different control strategies that has been carried to keep balance and to handle robot dynamics in humanoid robots.

2.1 Approaches in Humanoid Robot Control

Different methods has been used to make a robot move depending on the application, the complexity of the task, and even the specific nature of the robot. The main control methods used for the humanoid robots are as follows: Motion planning, Kinematic Approach, Dynamic Approach and Optimal Control. Each of the methods and its recent development are discussed below.

2.1.1 Motion Planning

Motion Planning as the name suggests a method where a robot automatically finds its desired or goal state from its initial configuration. For instance, consider a hand moving from it's current pose to another pose, motion planning allows to move to goal pose considering the presence of obstacles and consumption of time and energy. Currently, there are many applications in

industrial robots and mobile robots where the robot motion is planned within both structured and non-structured environment.

Even though, motion planning and its researches are progressing majorly within past decades, the recent approaches are combined with artificial intelligence, advantages of computer technology and mathematics. The main approaches can be found in books such as [13, 14]. An desirable concept in motion planning is *Configuration Space (CS)*, which is the set of all possible configurations for a robot to attain. For a robot with n independent degrees of freedom, *CS* is an n-dimensional manifold $\mathbb{M}$ that contains all the desired configurations $q \in \mathbb{M}$ of the robot. The importance of *CS* is that changes the problem of moving a body in *SE(3)* to moving a point in *CS*. The summary of this section is sourced from the literature work [2]. Then there exists,

- CS_{obs} is the *Obstacle Configuration Space* formed to generate self-collision or obstacle collision free set of configurations such that $CS_{obs} \in CS$.
- CS_{free} is the *Free Configuration Space* which holds the set of configurations for a freely roaming robot such that $(CS_{free} \bigcup CS_{obs}) \cap CS$.

Using these configurations, the problem of motion planning can be stated as finding the continuous path $p(t)$ through the desirable configurations from initial state $q(0)$ to goal state $q(f)$ avoiding collisions, that is $p : [0, 1] \rightarrow CS_{free}$ where t defines time parameterization.

Generic methods

The solution to motion planning problem can be processed through classical approaches using deterministic, sampling-based or path optimization algorithms. Each of the types are briefed below.

1. *Deterministic algorithms* - The deterministic algorithms are developed such that it computes the valid path everytime knowing almost all the variables of the environment. Methods such as *cellular decomposition, Voronoi diagrams, visibility graphs, potential fields and Canny's algorithms* rely on mathematical construction of the environment with the obstacles and provide CS_{obs}. Although these algorithms are complete, the computation of high-dimensional space is expensive and the environments are always not deterministic.

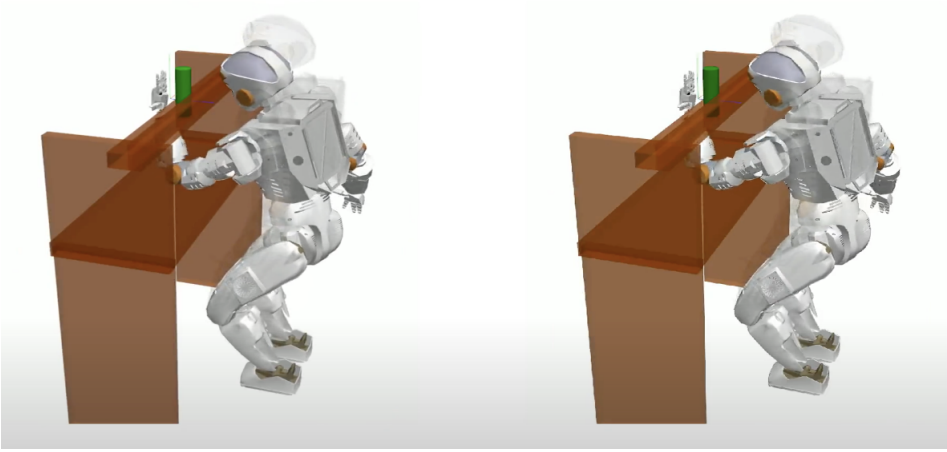

Figure 2.1: Sampling based motion planning of NASA Valkyrie [1]

2. *Sampling based algorithms* - These algorithms mostly approximate the connectivity of CS_{free} through random sampling configurations from CS and rejecting the configurations using boolean collision detection techniques. The main examples are *Probabilistic Maps and Rapidly-exploring Random Trees(RRT)* in combination with Voronoi diagrams promotes the obstacle avoidance configurations for a robot. The main advantage of sampling based algorithms are to handle the higher dimensional configuration space recovering a higher degree of completeness.

3. *Path optimization algorithms* - These algorithms provide optimization in terms of path planning and trajectory planning starting from a valid initial state to its goal position with the desired configurations. *Greedy optimization* tries to directly connect the start configuration to its goal state that generates a collision free shortest path by discretizing the path into n closest goal configuration relative to previous configuration.

Motion Planning in Humanoid robots

Classical motion planning techniques determine collision free trajectories considering only the geometric model of the robot. However the control of polyarticulated system needs the synthesis of robot models that describe the effect of joint variations on the whole robot configuration. In both the case, for instance considering an arm moving to it's goal position, the robot tends to make it more unusual, inefficient and unnatural movements. To overcome these problems, geometric models are replaced with kinematic models, dynamic models or optimal control and trajectories are generated. Additional constraints like multiple contacts and dynamic balance

of the system are considered. For instance, motion primitives that have been predefined by a human expert based on prior knowledge can be used to guide the planner [15].

In case of humanoid walking, the planner can be generated using deterministic approaches with dynamic alterations of the foot transition model considering the smooth transition of the trajectories for posture transitions. For these cases, sampling-based algorithms are considered to improve the degree of completeness. Either way, the higher dimensional configuration space is handled so that the problem is solved successively. An example is presented in [16] where a 36 degree of freedom robot is reduced to a 3 degrees of freedom bounding box and a PRM is applied for the path planning problem of the box. Another example is to present the constraints in the form of sub-manifolds of CS where a union of separate manifolds like contact limb position and static balance constraints can be used to plan the configuration space. In such cases, static balance control in humanoid robots and other legged robots can be obtained [17].

2.1.2 Kinematics Approach

Generally, *kinematics* is defined as a branch of science which deals with the study of the position, velocity and acceleration of a mechanical system without considering forces and the dynamic properties of the system (such as mass or inertia) that generate the motion. In humanoid and manipulator robots, the system is represented as rigid bodies composed of actuators and sensors. In contrast with the manipulators, the humanoid robots are not fixed to any environment and are highly mobile. This makes the humanoids (or humanoid robots) more redundant than the manipulators. This section briefs the state of the art and concepts of kinematic approach used in humanoid control.

Basic Concepts

The *joint space* is also called as *configuration space*, of a robot with n degrees of freedom (DoF) is a n-dimensional manifold Q containing all the possible joint values for a joint q can take. For humanoid robots, this space can be generalized to the operational points [18], which can represent any part of the body that may be of interest. In robotics, there exists four subdomains of kinematics namely, *(i) forward kinematics (or direct geometry), (ii) inverse kinematics (or inverse geometry), (iii) forward differential kinematics (or simply forward kinematics), and (iv)*

inverse differential kinematics (or simply inverse kinematics).

- **Direct Geometric Model:** For a robot with n DoF in a n-dimensional joint space such that $q \in Q$, there exist a pose $x \in SE(3)$ represented as

$$x = f(q)$$

described by a map $f : Q \to SE(3)$.

- **Inverse Geometric Model:** For a robot with n DoF in a n-dimensional joint space with $q \in Q, x \in SE(3)$, the joint space can be represented from a given pose for a certain operational point as

$$q = f^{-1}(x)$$

described by a map $f : SE(3) \to Q$. But there is a possibility for non-unique solution or non-existing solution (known as singularity).

- **Forward Kinematic Model:** For a robot with n DoF in a n-dimensional joint space such that $q \in Q, x \in SE(3)$, the operational twist $\xi \in SE(3)$ due to the joint variation $\dot{q}$ is described as

$$\xi = J(q)\dot{q}$$

Here, J is the basic Jacobian such that $J : T_q(Q)$ where $T_q(Q)$ is the tangent space of the Q space. Instead, the Jacobian J can be formulated analytically using the pose variation $\dot{x}$ and joint variation $\dot{q}$ as

$$\dot{x} = \frac{\partial x}{\partial q}\dot{q} \quad or \quad \dot{x} = J\dot{q}$$

then J is the task Jacobian.

- **Inverse Kinematic Model:** For a robot with n DoF in a n-dimensional joint space such that $q \in Q, x \in SE(3)$, finding the joint variations $\dot{q}$ that produce a pose variation $\dot{x}$ of the end effector as

$$\dot{q} = J^{-1}\dot{x}$$

and it can be solved iteratively.

Kinematic Control of Redundant robots

In general, the kinematic control involves a reference planner or trajectories $q_{ref}(t)$ in $SE(3)$ for which the robot joint trajectory $q(t)$ is evaluated against attaining the objective. Consider a task i in a robot with n DoF, then the Jacobian is of size $m \times n$ and the robot is redundant when $(n > m)$ [19]. The joint variations $\dot{q}$ can be represented if

- $(n > m)$. For task i, there exists a degree of redundancy $n - m$ for which the joint variation based on lease square method can be given by

$$\dot{q} = J_i^+ \dot{x}_i + (I_n - J_i^+ J_i) z_i$$

  where $J_i^+ = J_i^T (J_i J_i^T)^{-1}$ is the pseudo inverse of J_i, I_n is the identity matrix of size n and z_i is an n-dimensional arbitrary vector. The first term in the above equation is to minimize the norm solution while the second term is to find all possible solutions.

- $(n = m)$. The degree of redundancy is 0 and the joint variation can be defined as

$$\dot{q} = J^{-1} \dot{x}$$

  Note that the Jacobian here is a non-singular matrix.

- $(n < m)$. There doesn't exist any redundancy and no solution is found in this case.

For multiple tasks, the probability of finding a suitable solution decreases as the preceding tasks influence the current task, in other words, consider two tasks 1 and 2 for a redundant robot. There exist a solution for two tasks $x_1 = f_1(q)$ and $x_2 = f_2(q)$ with task priority for x_1 and x_2 respectively. First the variation q that solves the task according to the priority is determined from the differential equations by,

$$\delta x_1 = J_1 \delta q \tag{2.1}$$

$$\delta x_2 = J_2 \delta q \tag{2.2}$$

Then the joint variation δq for task 1 has infinitely many solutions and is given by

$$\dot{q} = J_i^+ \dot{x}_i + (I_n - J_i^+ J_i) z_i \tag{2.3}$$

Solving 2.3 and 2.2 for z_1 arbitrary vector,

$$z_1 = \hat{J_2^+}(\delta x_2 - J_2 J_1^+ \delta x_1) + (I_n - \hat{J_2^+}\hat{J}_2) z_2 \tag{2.4}$$

where $\hat{J}_2 = J_2(I_n + J_1^+ J_1)$ and $_2$ is an n-dimensional arbitrary vector. Then the solution of joint variation for the two priority tasks can be given by

$$\delta q = J_1^+ \delta x_1 + \hat{J_2^+}(\delta x_2 - J_2 J_1^+ \delta x_1) + (I_n - J_1^+ J_1)(I_n - \hat{J_2^+}\hat{J}_2) z_2 \tag{2.5}$$

Generalized form of Task Priority

Kinematic Control in humanoid robots

Humanoid robots as a rigid body representation, from a kinematic point of view present a tree-like structure that includes multiple connected chains, and a high number of DoF. This results in higher degree of redundancy with respect to most tasks, in which case the robot is said to be under-constrained. Therefore, methods developed for generic redundant robots are usually applied in humanoid robotics with some adaptations or additions. Due to the complexity of the kinematic configuration, closed-form solutions for the IK problem are usually very complex, but recently some classical methods have been used and modified, to obtain closed equations for specific humanoid robots which are treated as a composition of several kinematic chains. However, methods based on instantaneous IK, which compute an increment in q, are usually preferred. This linearization of the problem offers an infinite number of feasible solutions for humanoid robots: there exist different joint updates that achieve the same task. This leads to the possibility of performing different tasks at the same time, and the IK control must be capable of properly handling them.

Methods based on task-prioritization solve these problems and have become the preferred techniques in IK control. They can even be considered as the current "state of the art" in humanoid robotics control due to their relatively low computational cost, their straightforward implementation, and the maturity of the approach. Several works with different humanoid

platforms use this methodology to solve the redundant IK problem at the velocity level using only equality constraints [20, 21, 22]. For imposing inequality constraints to the control framework at any hierarchical level, a sequence of optimal resolutions for each priority level has been proposed in [23], a more efficient computation based on orthogonal decompositions can be found in [24] and a smooth interchange between priority of consecutive prioritized tasks is introduced in [25].

2.1.3 Dynamics Approach

Dynamics is the study of the relation between the robot motion and the generalized forces that act on the robot generating that motion. This relation considers parameters such as lengths, masses and inertia of the elements composing the robot.

Basic Concepts

In the dynamic model, the motion is represented through joint variables acceleration $\ddot{q}$, or operational points acceleration $\ddot{x}$. For rotational joints, the generalized forces are equivalent to the joint torques, and for prismatic joints they are the joint forces. There are two main problems in dynamics:

- **Forward Dynamics:** It expresses the motion of the robot as a function of the generalized forces applied to it.
- **Inverse Dynamics:** It expresses the generalized forces acting on a robot as a function of the robot motion.

There exist two main formulations to compute the robot dynamic model: the Lagrange approach, and the Newton-Euler approach. The Lagrange approach's main advantage is the clear separation of each component of the model; but in general, it is computationally expensive. The Newton-Euler approach does not provide a clear separation of the terms but due to its recursivity a lower computation time can be obtained. Thus, it is the preferred implementation for computer calculations. The most used algorithms for this approach can be found in [26, 27]. The Lagrange method and Newton Euler's method are detailed below

1. **Lagrange Formulation**

The Lagrange formulation descibes the behavious of a dynamic system in terms of work and energy stored in the system. The Lagrange equations are written in the form:

$$\tau = \frac{d}{dt}(\frac{\partial L}{\partial \dot{q}})^T - (\frac{\partial L}{\partial q})^T \tag{2.6}$$

where τ is the generalized forces on the system, q is the joint vector, $\dot{q}$ represent the joint velocities and $L = K - U$ is the Lagrangian function with K as kinetic energy and U as potential energy respectively.

The kinetic energy K for body i is expressed in the form,

$$K_i = \frac{1}{2}\int v_{m_i}^T v_{m_i} dm \tag{2.7}$$

where $v_{m_i} = v_i + \omega_i \times \vec{r_{m_i}}$ is the velocity of point m_i on the body i which can be expressed as a function of twist, $\xi = [v_i^T, \omega_i^T]$

The potential energy U for body i is defined as,

$$U_i = -m_i \mathbf{g}^T \vec{r_{CoM_i}} \tag{2.8}$$

where $\mathbf{g}$ is the gravity vector and $\vec{r_{CoM_i}}$ is the distance vector to CoM of body i. The equation 2.6 can be put in a matrix form as

$$\tau = M(q).\ddot{q} + c(q, \dot{q}) \tag{2.9}$$

where $M(q)$ is the generalized robot inertia matrix and $c(q, \dot{q})$ is the vector of Coriolis, centrifugal and gravity effects. This model is called as inverse dynamic model.

2. **Newton Euler Formulation**

Newton Euler (NE) equations allow computation of the sum of external forces $\sum f_j$ and moments $\sum m_{CoM_i}$ (including the gravity effects) is represented as

$$\begin{aligned} \sum f_i &= m_i \dot{v_{CoM_i}} \\ \sum m_{CoM_i} &= \mathbf{I}_{CoM_i}\dot{\omega_i} + \omega_i \times (I_{CoM_i}.\omega_i) \end{aligned} \tag{2.10}$$

where $\dot{v_{CoM_i}}$ is the acceleration of the CoM of body i; $\dot{\omega_i}$ is the angular acceleration of the body i and $\mathbf{I_{CoM_i}}$ is the inertia matrix of body i. These equations are usually recursive algorithms [27, 26] which make the computation less expensive when a computer involves.

The introduction of centroidal momentum and centroidal dynamics based on the Centre of Mass (CoM) of a system is particularly a import concept since the humanoids build large momenta(so far only angular momentum is larger and linear momentum on legger humanoids are not upto the mark). The dynamic model of a robot can be expressed in two ways depending on the spaces that are used to describe the motion and the control input. The two approaches are: the joint space formulation, and the operational space (or task space) formulation. The joint space formulation is the classical approach and uses the joint space acceleration to specify the motion, and the generalized forces acting on the actuated joints to describe the control. The operational space formulation represents the motion directly using the task space acceleration, which needs a reformulation of the forces as task space generalized forces.

Dynamic Control in Humanoid Robots

For a humanoid robot, classical dynamic control techniques are not sufficient since coordination of the motion is required, environmental forces need to be considered, and balance has to be kept at all time. For the robot balance, a stability criterion such as keeping the CoM or the ZMP inside the support polygon must be enforced. For planar surfaces, the constraint on the ZMP implies a control of the contact forces. Thus, the interaction with the environment is always present since the feet are in contact with the ground and additional contacts of other parts of the robot body might also be necessary. These contacts generate an effect on the joints generalized forces, which cannot be controlled using only kinematic methods but with dynamic approaches. The dynamic control ensures the physical feasibility of the motion and allows for faster movements without losing balance.

The Operational Space Inverse Dynamics (OSID) is a more specific framework for controlling

the whole-body of humanoid robots considering contacts and a set of different constraints. This approach proposed in [28, 29] is based on a two stage mapping to obtain consistent contact forces and is equivalent to successive projections onto the nullspaces of the previous tasks. Therefore, new tasks can be added without dynamically interfering with higher priority tasks. Other methods use the OSID within frameworks that involve some type of optimization to find the local solution. The most popular approaches use Quadratic Programming (QP), which allows for the specification of both equality and inequality constraints. The latter type of constraints is fundamental in humanoid robotics to directly model unilateral contacts, and it is also important to properly specify some particular tasks. Although the previous approaches exploit the full robot dynamics, the angular momentum is not explicitly controlled within these frameworks. Nevertheless, it has been shown that the angular momentum is a natural and important part of human motion, specially when performing complex and fast movements [30].

2.1.4 Optimal Control

Optimal control, also known in robotics as trajectory optimization or trajectory filtering, consists in finding a trajectory and its associated control law (policy) that satisfies some predefined optimality criterion. In general mathematical terms, it concerns the properties of control functions which, when inserted into a differential equation, give solutions that minimize a cost or measure of performance. But it also concerns optimization problems with dynamic constraints which might be functional differential equations, difference equations, partial differential equations or equations with another form. This section is summarized from the literature work [2] which provides a brief overview of optimal control in humanoid robots.

Optimal Control in Humanoid Robots

In robotics, optimal control can be used to find the trajectories from an initial posture to a final desired posture, specified as a whole or as a set of sub-objectives, satisfying certain constraints. Very fast and powerful movements can be generated with optimal control, which can comprehend the problems of inverse kinematics or inverse dynamics and can therefore produce better movements. The problem with IK and OSID alone is their inability to properly handle the CoM accelerations, thus overrestricting the motion. A solution typically relies

on a dedicated submodel, like a linearized inverted pendulum [31] to capture the future of the system, but this ad-hoc resolution increases the control architecture complexity. Thus, using these schemes the future states of the system can be somehow predicted, but dedicated submodels would need to be developed for each case. However, optimal control is the most suitable approach that allows to take into account all the constraints at the same time. In fact, optimal control can automatically generate the proper trajectory for the CoM in order to achieve fast movements: it acts as a classical pattern generator used for walking schemes, but additionally incorporating whole-body motion. A serious challenge to optimization-based approaches in robotics is that the timescales of the dynamics are faster than in other applications and need a faster response. Currently, the main problem is the computational time, due to the high number of DoF, that forbids its use in real-time applications.

The main drawback of optimal control is the curse of dimensionality, which is particularly important for humanoid robots whose state space is so large that no control scheme can explore all of it in advance and prepare suitable responses for every situation. It would be desirable to obtain optimal control in real time; however, currently there is no approach that can achieve this: the solutions are very time consuming and generic solvers tend to get stuck into local minima or they even return trivial solutions. The problem of finding the proper formulation and resolution of optimal control is still an open issue in robotics.

2.2 Approaches in Humanoid Balance Control

There has been various control strategies that has been implemented on the humanoid robot for various tasks but, the balance control strategies are the dominant task when it comes to legged humanoids. The balance strategies are mostly based on the centroidal momentum and centroidal dynamics for the complex structured humanoids. These single point mass systems are less computational expensive considering point approximation method of Centre of Mass (CoM). Note that the concepts of CoM and ZMP will be discussed later in chapter 3. The balance control approaches based on single point masses will be discussed in this section among which the most common methods are

1. Linear Inverted Pendulum Model

2. Double Inverted Pendulum Model

3. Spherical Inverted Pendulum Model (Simple and Double)

2.2.1 Linear Inverse Pendulum Approach

When it comes to real-time motion planning and control, single point masses plays extremely major role in holding the balance of the biped robots. A number of physical points can be simplified and summarized from these models where some of the points include Centre of Mass (CoM) and Centre of Pressure (CoP). The CoM is defined as the sum of the mass of the individual links of a system and CoP is defined as the point on the ground that represents the overall force interaction of all ground contact points. The dynamics of any system of rigid bodies can be approximated usind the dynamics at CoM. At any time t, the sum of the force acting on the system results in the acceleration of CoM in the system. Mathematically, it can be represented as,

$$S. \begin{bmatrix} f_{gr} \\ \tau_{gr} \end{bmatrix} = \begin{bmatrix} ma \\ \dot{H} \end{bmatrix} \tag{2.11}$$

where m is the total mass of the system, a is a (3×1) vector representing the CoM acceleration. $\dot{H}$ represent the change in amgular momentum, f_{gr} is the ground reaction forces and τ_{gr} is the torque vector. And S can be represented as the Selection Matrix equal to [32],

$$S = \begin{bmatrix} I_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ (P_R - COM)\times & (P_L - COM)\times & I_{3\times3} & I_{3\times3} \end{bmatrix} \tag{2.12}$$

where P_R and P_L are the feet position, $I_{N\times N}$ and $0_{N\times N}$ are the identity matrix and zero matrices of degree N; and $v\times$ is the left cross product matrix of any vector v. Equation 2.11 represents the forces acting at CoM due to gravity and ground reaction forces in the first row and torque about CoM due to change in anguluar momentum.

Consider a simple inverted pendulum model as in figure [2.2] with single point mass m and link length l, given the assumptions that no angular momentum will be generated or changed for the single point mass. Furthermore, an additional assumption is made to the dynamics of the system can be represented as,

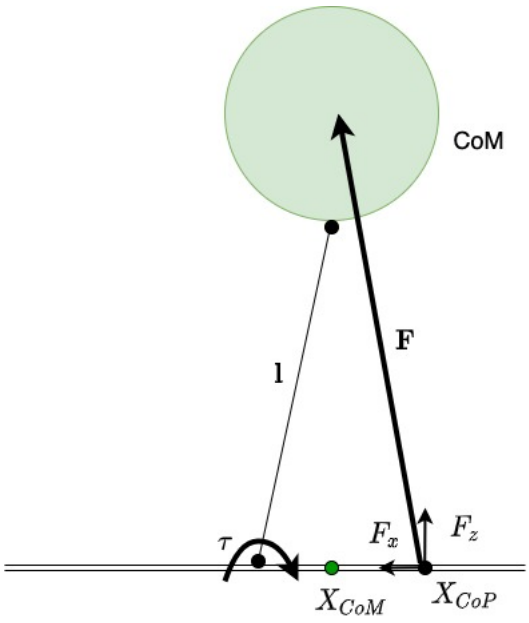

Figure 2.2: Linear Inverse Pendulum Model

$$m\ddot{x} = \frac{m\mathbf{g}}{z_0}(x - x_{CoP}) \tag{2.13}$$

where x represents the CoM horizontal position at a constant height z_0, and $x_{CoP} = \frac{F_T - \tau}{F_N}$ represent the CoP position where F_T is the tangential component of the ground reaction force, τ is the ankle torque and $F_N = m\mathbf{g}$ is the normal vertical component of the ground reaction force. Thus the LIPM model can be used for ankle balance strategy for legged and humanoid robots.

There are number of ways of using X_{CoP} as the control input for the system as surveyed in [32]. Firstly, the rate of change of x_{CoP} can be limited to achieve smooth trajectories. Secondly, the CoP position is often a measured quantity in most humanoid robots. Thirdly, it is not possible to change CoP position instantaneously. except during the transition from single to double support.

2.2.2 Double Inverse Pendulum Approach

The single point mass based assumptions from the previous section 2.2.1 can cause internal forces disturbing the system. This is due to the fact that the primary assumption is massless legs of the system. These systems are linearised models of the good approximation of the system dynamics.

Consider a double inverted pendulum model as in figure 2.3 with mass m and link lengths l_1 and l_2, to capture the effect of the upprt body motion of the biped balance, the humanoid robot can be modelled as non-linear fully-actuated, unconstrained double inverse pendulum mode. Then the equation of motion s can be given by,

$$M(q)\ddot{q} + c(q, \dot{q}) = \tau \tag{2.14}$$

where q is the joint vector, M is the mass-inertia matrix, τ is the torque vector and c is the vector of a Corolis, centripedal and gravitational forces. The DIPM model used the hip-balance strategy defined in [33]. In these models, the feet are normally assumed to be in contact with ground and do not contribute kinetic or potential energy of the system. The CoP calculation is still the same assuming that it does not leave the support polygon defined by the feet as in equation 2.13.

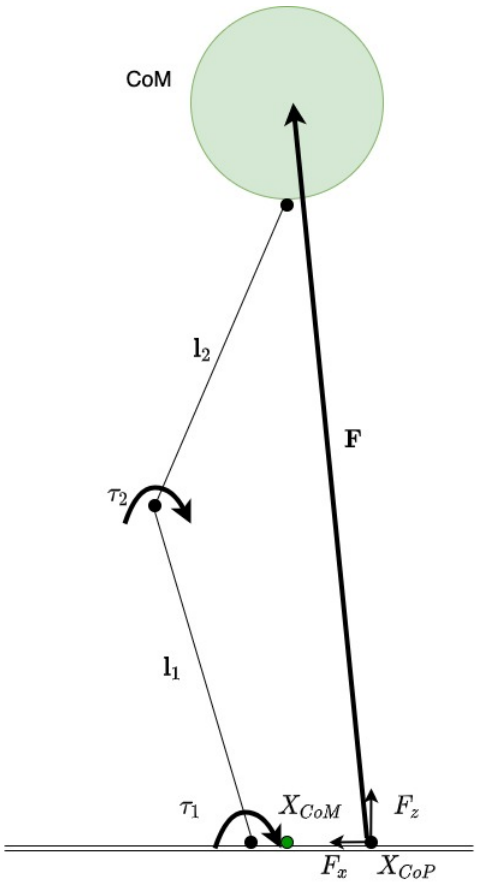

Figure 2.3: Double Inverse Pendulum Model

2.2.3 Spherical Inverse Pendulum Approach

There are evidences to use simplified non-linear spherical inverse pendulum model for defining the motion of the humanoid robots. These models are marginally stable and if it is subjected to any slight disturbance, it will lose its upright stable posture. To maintain this unstable equilibrium of the SIP model, different forms of feedback controls are used. The equations of motion for non-linear spherical inverse kinematic models are given by [32],

$$\begin{aligned} \ddot{\theta} &= \frac{\tau_\theta}{ml^2} + \frac{\mathbf{g}}{l}\cos(\phi)\sin(\theta) - \phi^2\sin(\theta)\cos(\theta) \\ \ddot{\phi} &= \frac{\tau_\phi}{ml^2\cos^2(\theta)} + \frac{\mathbf{g}\sin*(\phi)}{lcos(\theta)} + 2\frac{\dot{\phi}\dot{\theta}\sin(\theta)}{\cos(\theta)} \end{aligned} \tag{2.15}$$

where θ and ϕ represent the ankle orientation along x- and z-axis respectively.

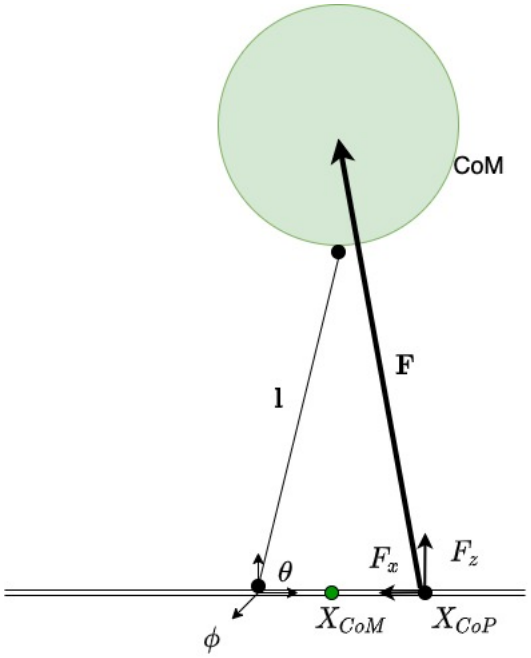

Figure 2.4: Linearized Spherical Inverse Pendulum Model

From equations 2.15, it can be seen that the system is non-linear and there is a degree of cross coupling between the two degrees of freedom. Like any other LIPM model, simple SIPM models use ankle-balance strategy. To use traditional control methods, the system needs to be linearized for small values of θ and ϕ. Then the equation of motions describing the pendulum are represented as,

$$\ddot{\theta} = \frac{\tau_\theta}{ml^2} + \frac{\mathbf{g}}{l}\theta$$
$$\ddot{\phi} = \frac{\tau_\theta}{ml^2} + \frac{\mathbf{g}}{l}\phi \tag{2.16}$$

Equations 2.16 are the linearized approximation of the SIPM model which holds the dynamics of the system with $x_{CoP} = -l \times \sin(\theta)$ as,

$$m\ddot{x} = \frac{m\mathbf{g}}{z_0}(x - x_{CoP}) \tag{2.17}$$

2.2.4 Static vs Dynamic Balance

During the walking gait, at any time, the projection of the centre of mass if the robot is within the support polygon [34]. This allowed the static walker to maintain balance if stopped at any timestep. However, this resulted in the slower walking speeds and larger translation of the centre of mass from one foot to other. The effect of the inertial forces acting on the robot segments are not taken into account except for the gravitaional forces.

During dynamic balance, the centre of mass is allowed outside the support polygon to an extent but there might be parts where the robot might fall [34]. The velocity and acceleration of each link of the robot model is taken into account resulting in comparitively faster motion maintaining the postural stability. Most of the modern robots are using dynamic control for its efficiency and versatile gait structure compared to static control.

2.3 Approaches in Dynamic Motion Retargetting

This section presents the various works focusing on dynamics based motion retargetting systems implemented on real robots.

2.3.1 Dyanmic Retargetting in HRP-2 robot

The case study presented in [2] for motion imitation focusing on dynamics of human motion allows quickly and efficiently to generate long sequence of dynamic movement for humanoid robots. The motion generation is based on combination of two very efficient tools: motion

capture and hierarchical operational space inverse dynamics, which enables the retargetted dynamics to fit with the constraints of the real robot. The method produces reliable movements on the robot.

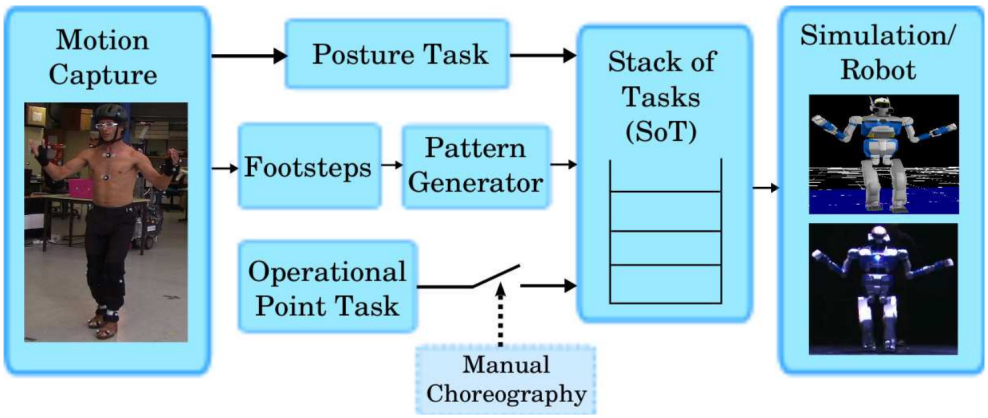

Figure 2.5: Dynamic Retargetting Scheme [2]

From this study, it can be considered that the operational-space inverse-dynamics is a mature tool that is able to replace the inverse kinematics approach for all robots where dynamics matters. Particularly, in humanoid robotics, inverse dynamics can be directly applied by sending the reference acceleration output, obtained with the inverse-dynamics solver, to the robot. In particular, the Stack of Tasks (SoT) proposed in [2], while enforcing the dynamics of the retargeted motion, provides the robot programmer with some easy edition capabilities to correct the defects of the retargeted motion or to augment the original movement with some artificial features.

2.3.2 Motion Retargetting in iCub

The work proposed in [3] presebted a framework for teleoperation of iCub robot based on inverse kinematics with Quadratic solver. It allows a robust real-time retargetting of generic motions. The CoM retargetting and fall detection proposed in this work is robust due to the presented ZMP correction approach that guarantees the stability of the retargeted motion in double support. This approach is validated in simualtion and on real robot perfecting the balance restoration in the kinematic approach. More information on this work is detailed and adapted in chapter 3.

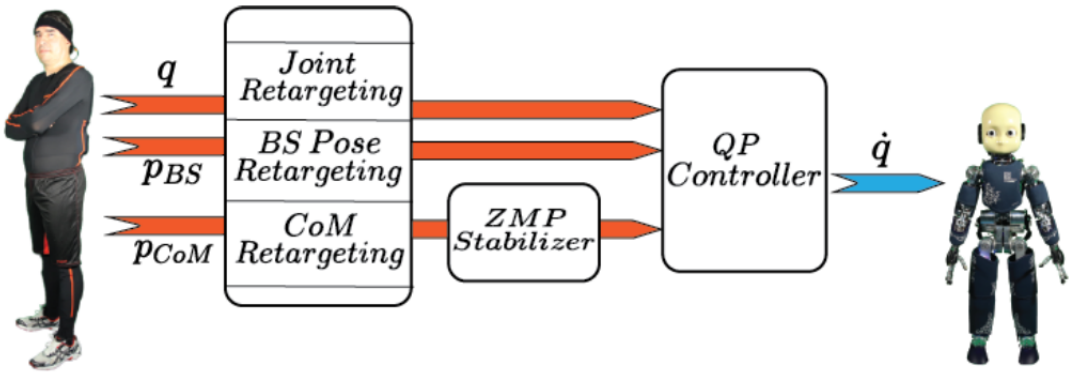

Figure 2.6: Motion Retargetting Pipeline [3]

2.3.3 Hybrid Retargetting using Time Scaling

This work by [35] is to take advantage of motion tracking by manipulating the motion by time scaling. This method changes the traveling time of the reference motion by decelerating the motion if unbalanced and by accelerating the motion in order to synchronise with the reference motion by decelerating the motion if unbalanced and by accelerating inorder to keep it synchronising with the human actor. This approach serves as a different variant in the domain of motion imitation of humanoid robots since when a suitable solution with time scaling is not found, joint based control to ensure the balance of the system is turned back on. A set of motions has been recorded and are validated in simuation using virtual NAO robot.

Chapter 3

Dynamics Based Whole Body Imitation

Whole body motion imitation of a humanoid robot is a challenging problem due to the kinematic and dynamic complexity of the robot. Humanoids are highly redundant but loses a higher number degrees of freedom compared to human actor. Mapping the actions and poses performed by human actor is kinematically achieved [36, 37] even though the motion mapped here are slow paced and non-complex actions. Though the high paced actions are achieved and imitated on computer graphics [38], the problem on humanoid robots are completely different domain. To perform face paced actions, dynamics parameters of the robot model need to be taken into account for which only a few solutions are proposed [2, 3]. As an additional contribution to the problem, this chapter explains a few concepts and methods to improve the imitation pace using acceleration control keeping the dynamic balance of the robot.

3.1 Dynamic Considerations

The dynamic model of the robot states the relation between the generalized torques τ and acceleration $\ddot{q}$ of the robot given its dynamic parameters like Mass and Inertia. Since the humanoids are not attached to the environment, the feet and other humanoid parts like hand (even though an assumption is made that only feet are in contact with the environment for this work) impose additional constraints that needed to be taken cared for keeping the motion intact and balance at all the time. Thus the representation of the robot must include these constraints in addition to the balance constraints for which the control strategy must satisfy at all cost. The dynamic model of the humanoid robot (focused towards NAO robot) is discussed

in this section.

3.1.1 Rigid body dynamics

The Newton-Euler's equations for a rigid body $\mathcal{B}_j$ that are related to linear momentum p_j and angular momentum ω_j is formulated as follows.

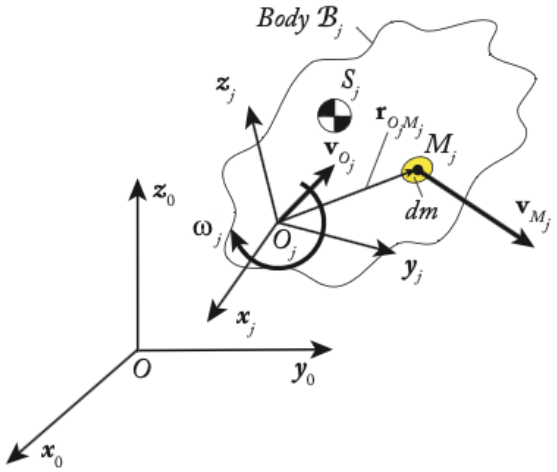

Figure 3.1: Schematic of a rigid body

From equations 2.10, the sum of forces and sum of moments at CoM are given by,

$$\begin{aligned} \sum f_j &= \frac{d}{dt}|_{\mathcal{F}_j} p_j = m_j \dot{v_{CoM_j}} \\ \sum m_j &= \frac{d}{dt}|_{\mathcal{F}_j} (\mathbf{I}_{CoM_j}\omega_j) = \mathbf{I}_{CoM_j}\dot{\omega_j} + \omega_j \times (I_{CoM_j}.\omega_j) \end{aligned} \tag{3.1}$$

where $\dot{v_{CoM_j}}$ is the acceleration of the CoM of body j; $\dot{\omega_j}$ is the angular acceleration of the body j and $\mathbf{I_{CoM_j}}$ is the inertia matrix of body j. The NE equations can also be expressed at the origin $\mathcal{O}$ to the body $\mathcal{B}_j$ can be expressed as proposed in [39] as

$$\begin{aligned} \sum f_j &= m_j \dot{v_j} + \dot{\omega_j} \times ms_j + \omega_j \times (\omega_j \times ms_j) \\ \sum m_j &= \mathbf{I}_{\mathcal{O}}\dot{\omega_j} + \omega_j \times (I_{\mathcal{O}}.\omega_j) + ms_j \times \dot{v_j} \end{aligned} \tag{3.2}$$

where $ms_j = m_j.r_{\mathcal{O}CoM_i}$ is the vector of first moment of inertia and $\mathbf{I}_{\mathcal{O}}$ is the inertia matrix at origin $\mathcal{O}$. Using the screw notation, equation 3.2 can be rewritten as,

$$\sum w_j = \begin{bmatrix} \sum f_j \\ \sum m_j \end{bmatrix} = \begin{bmatrix} m_j \mathcal{I}_3 & m\hat{s}_j{}^T \\ m\hat{s}_j{}^T & I_{\mathbf{O}} \end{bmatrix} \begin{bmatrix} \dot{v}_j \\ \dot{\omega}_J \end{bmatrix} + \begin{bmatrix} \omega_j \times (\omega_j \times (\omega_J \times ms_j)) \\ \omega_j \times (I_{\mathcal{O}} \omega_j) \end{bmatrix} = M_j \dot{q} + c_j \quad (3.3)$$

where $\mathcal{I}_3$ is the identity matrix of size 3, M_j is the generalized inertial matrix of body $\mathcal{B}_j$ and c_j is the vector of Coriolis and centrifugal effects. $\sum w_j$ (alias Γ) is the wrench representation which holds the sum of forces and moments and the equation 3.3 represent the direct dynamic model for a system. Then the inverse dynamic model can be given by,

$$\dot{q} = M_j^{-1}(\Gamma - c_j) \quad (3.4)$$

3.1.2 Multi body dynamics

For modelling multi-body dynamics or tree-structured systems in this case, recursive NE algorithm is the most efficient problem than the recursive Lagrangian equations [40].

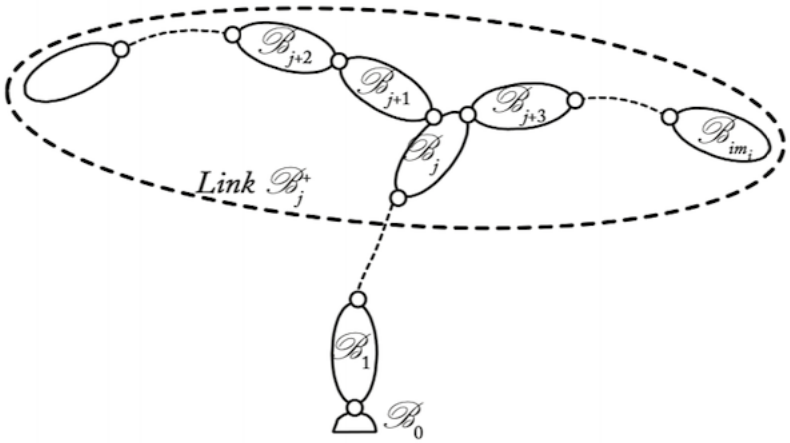

Figure 3.2: Rigid-body representation of tree structured model

The recursive algorithm for multi-body system adapted from equation 3.2 is given by [40]

$$\begin{aligned} \sum f_i &= f_i - \sum_r f_r + m_i \mathbf{g} - f_e \\ \sum m_i &= m_i - \sum_r (R_k m_k + r_k \times f_k) + ms_i \times \mathbf{g} - m_e \end{aligned} \quad (3.5)$$

where $r_k = r_{\mathcal{O}k}$ is the distance vector from origin to force points on body $\mathcal{B}_i$; $ms_i \times \mathbf{g} = r_{\mathcal{O}CoM} \times m_i\mathbf{g}$ f_r and m_r are the reaction forces and moments respectively exerted by the body $\mathcal{B}_r$ on the body $\mathcal{B}_i$ at point $\mathcal{O}_r$; f_e and m_e represent the forces and moments exerted on the environment by body $\mathcal{B}_i$. These values are assumed to be known.

3.1.3 Dynamic Model of NAO robot

This section derives the dynamic model of the humanoid robot NAO considering the absence of force sensors in the robot feet. The mass, CoM and Inertial matrices of the NAO robot v5 H25 is documented on the aldebaran website [5]. The mass and coordinates of CoM of individual links are taken and processed. The CoM given in the documentation is presented relatively to a local coordinate system $\mathcal{R}_j$ attached to the corresponding body $\mathcal{B}_j, i = 1, 2, ..., 25$. To define the frame system corresponding to the modified DH rule, one has to rotate, translate or do both to compensate the new model for inverse dynamics model.

Let $CoM_j^A = [x_A, y_A, z_A]^T$ be the coordinates of the local reference frame $\mathcal{R}_j$ specified in the documentation [5]. Then the inertial tensor considering the system of particles of the body about the center of mass with $r = xyz$ as the position vector is defined as,

$$J_j^j = \begin{bmatrix} \int(y^2+z^2)dm & -\int xydm & -\int xzdm \\ -\int xydm & \int(x^2+z^2)dm & -\int yzdm \\ -\int xzdm & -\int yzdm & -\int(x^2+y^2)dm \end{bmatrix} \tag{3.6}$$

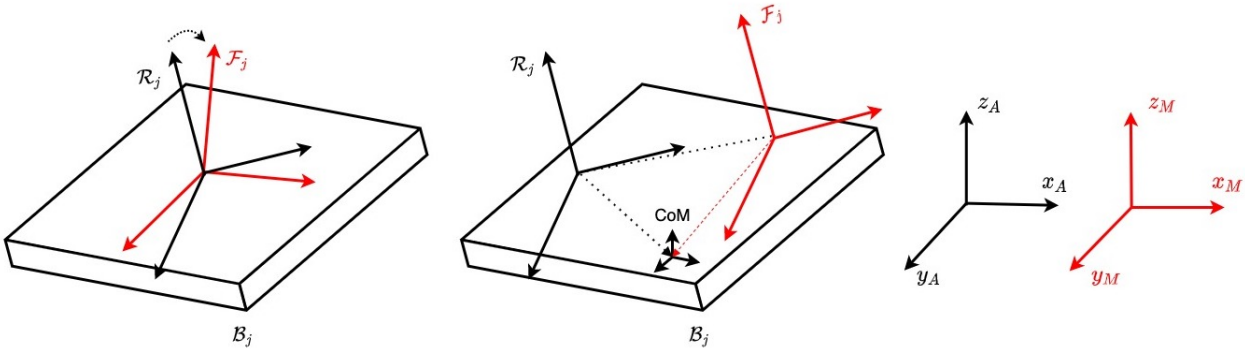

Figure 3.3: Rotation and Translation of Body reference frame

Since the matrix in equation 3.6 depends on the coordinates of the CoM, when the reference

frame changes, the value of inertial tensors changes automatically [35]. To perform rotation from the provided CoM coordinate CoM_j^A represented from $\mathcal{R}_j$ to the coordinate CoM_j^M represented in $\mathcal{F}_j$, the rotation matrix is defined and the rotation matrix can be computed as,

$$^{A}J = {}^{M}A_{A}\,{}^{A}J({}^{M}A_{A})^{T} \tag{3.7}$$

and the translation can be computed by [35],

$$^{M}J + m\begin{bmatrix} y_A^2+z_A^2 & -x_Ay_A & -x_Az_A \\ -y_Ax_A & x_A^2+z_A^2 & -y_Az_A \\ -z_Ax_A & -z_Ay_A & x_A^2+y_A^2 \end{bmatrix} = {}^{A}J + m\begin{bmatrix} y_M^2+z_M^2 & -x_My_M & -x_Mz_M \\ -y_Mx_M & x_M^2+z_M^2 & -y_Mz_M \\ -z_Mx_M & -z_My_M & x_M^2+y_M^2 \end{bmatrix} \tag{3.8}$$

Once the frames are transformed to the respective notation, recursive Newton-Euler formalism of dynamic equation from section 3.1.1 can be used to formulate the dynamic model of the equations.

3.1.4 Zero Moment Point (ZMP)

The zero moment point (ZMP) of the legged system can be defined as a point where the reaction force at the contact of the foot with the ground doesn't produce any horizontal moment i.e., the point where the total of horizontal inertia and gravity forces equals zero. ZMP is useful in defining the stability criterion of the legged system using the support polygon of the foot. A support polygon is a region of perpendicular projection of the balancing end-effectors (legs) that carry the robot's weight. A ZMP is a projection of the CoM onto the support polygon and it should lie within the region. Any attempt for the ZMP outside the support polygon to an extent will result in robot fall. Having these considerations, the ZMP of a biped system can be given by,

$$P_{ZMP} = P_{CoM} - \ddot{P}_{CoM}\left(\frac{z_{CoM} - z_{ZMP}}{\mathbf{g} + \ddot{z}_{CoM}}\right) \tag{3.9}$$

where $\mathbf{g} = -9.81 m/s^2$ is the gravity acceleration. Fixing these points onto a specific position on the floor is over-constraining and leaves less free DOF for the other tasks. However, this precise control over the point positioning is advantageous for choosing safer positions far from the foot's edges and for avoiding points too far from the ankle which would require too much torque to hold the whole body [6].

3.2 Control Approach

Proposing a control approach for a humanoid robot imitating the human actor differs from standard humanoid control approaches since feedback from both human and humanoid robot needs to be taken into account. This section explains the concepts and the approach carried out for humanoid control for motion imitation using Stack of Tasks (SoT).

3.2.1 Balance Control

CoM Retargetting

To track the Centre of Mass (CoM) of the human actor, an implementation from [4] is adapted to track the normalized offset on human's CoM relative to the support feet. For this case, a projection on 2D plane is considered. The human CoM P_{CoM}, H can be expressed using modified Hanavan Model approximation. The normalized offset $\mathbf{o}$ between 0 and 1 can be computed as follows.

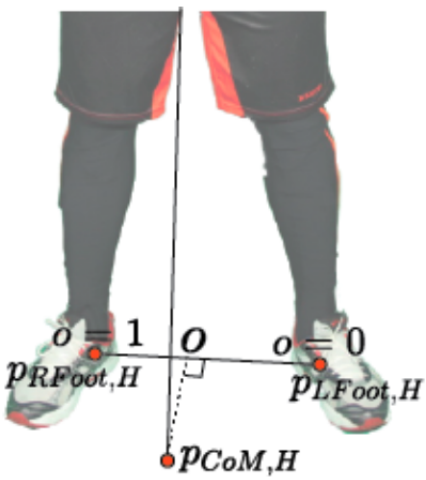

Figure 3.4: Representation of offset projection of human actor [4]

$$\mathbf{o} = \frac{(P_{CoM,H}) - P_{LFoot,H})(P_{RFoot,H} - P_{LFoot,H})}{||P_{RFoot,H} - P_{LFoot,H}||^2} \tag{3.10}$$

where $P_{CoM,H}$ represent the position of CoM projection of human actor. on the horizontal plane; $P_{LFoot,H}$ and $P_{RFoot,H}$ are the position of the left and right foot of the human actor respectively. The normalized offset $\mathbf{o}$ has an value of 0.5 during double support and has 0 or 1 during single support. The robot CoM projection then is calculated as,

$$P_{CoM,R} = P_{LFoot,R} + \mathbf{o}(P_{RFoot,R} - P_{LFoot,R}) \tag{3.11}$$

To retarget also changes of the human CoM that are not on the line connecting the two feet, we first measure the maximum backward and forward CoM displacement of the human and of the robot over their support polygon (with the origin lying on the feet line), i.e $\delta_{CoM_{back},H}, \delta_{CoM_{forw},H}, \delta_{CoM_{back},R}$ and $\delta_{CoM_{forw},R}$ respectively. Then the retargetting the human CoM displacement $\triangle_{CoM,H}$ within the range such that $-\delta_{CoM_{back},H} \leq \triangle_{CoM,H} \leq \delta_{CoM_{forw},H}$ can be computed as ,

$$\mathbf{o}' = \frac{(\triangle_{CoM,H} - (-\delta_{CoM_{back},H}))}{(\delta_{CoM_{forw},H} - (-\delta_{CoM_{back},H}))} \tag{3.12}$$

for which the robot CoM displacement is

$$\triangle_{CoM,R} = \mathbf{o}'(\delta_{CoM_{forw},R} + \delta_{CoM_{back},R}) - \delta_{CoM_{back},R} \tag{3.13}$$

This displacement is then used in the orthogonal direction of the line connecting the two feet of the robot.

Floating base Control

To control the height of the floating base of the robot,, the pelvis point of the human actor is considered. The deviation of the pelvis point of the human is given by,

$$\triangle_{base_{t,H}} = base_{t,H} - base_{0,H} \tag{3.14}$$

where t is the timestep such that $t \geq 0$. Then the correction for robot base is given by,

$$\triangle_{base_{t,H}} = \frac{h_{base,R}}{h_{base,H}} \triangle_{base_{t,R}} \tag{3.15}$$

where $\alpha = \frac{h_{base,R}}{h_{base,H}}$ is the ratio of height of the floating base of the robot and of the pelvis of the human, at N-pose. Then the height of the robot base at each timestep can be calculated by,

$$base_{t,R} = base_{0,R} + \triangle_{base_{t,R}} \tag{3.16}$$

The change of orientation of the floating base is also calculated in a similar way, by computing the roll, pitch and yaw from the quaternion information given by the motion capture system.

ZMP Retargetting

During whole body teleoperation of humanoid robots, disastrous crashes may occur if the desired CoM trajectories recorded from the human do not ensure the balance of the controlled robot when retargeted.

To this scope, we propose a QP-based "preprocessor" that adjusts in real-time the desired commanded CoM to satisfy constraints that represent a condition for dynamic balance. In order to achieve a stable CoM trajectory we employ the linear inverted pendulum model (LIPM) in combination with the Zero Moment Point (ZMP) criterion. The ZMP is represented with a point on the ground plane where the tipping moments, generated by the gravity and the inertial forces, are equal to zero. A humanoid robot keeps its balance if the ZMP is contained inside the support polygon of the robot.

Through the LIPM model it is possible to establish a simple relation between the ZMP and the CoM dynamics:

$$\ddot{p}_{CoM} = \frac{g}{h}(p_{CoM} - p_{ZMP}) \tag{3.17}$$

3.2.2 Posture Control

This section provides a method for the simplification of humanoid robot and the human actor as a composition of inverted pendulum models. These simplified multi-double inverted pendulum models are used for posture control during the motion retargetting.

Multi-Double Inverted Pendulum Model (M-DIP)

The full humanoid robot may be simplified using a combination of different double inverted pendulums as represented in figure 3.5. The current position of the individual pendulums is represented as $\mathbb{P}_i$ and $\mathbf{P}_i$ for human actor and humanoid respectively where i represent the left hand (lh), right hand (rh), right leg (rl) and left leg (ll). Then the state of the human actor $\mathbb{P}$ and humanoid robot $\mathbf{P}$ is represented as,

$$\mathbb{P} = \begin{bmatrix} \mathbb{P}_{rh} & \mathbb{P}_{rl} & \mathbb{P}_{lh} & \mathbb{P}_{ll} \end{bmatrix}^T \qquad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{rh} & \mathbf{P}_{rl} & \mathbf{P}_{lh} & \mathbf{P}_{ll} \end{bmatrix}^T$$

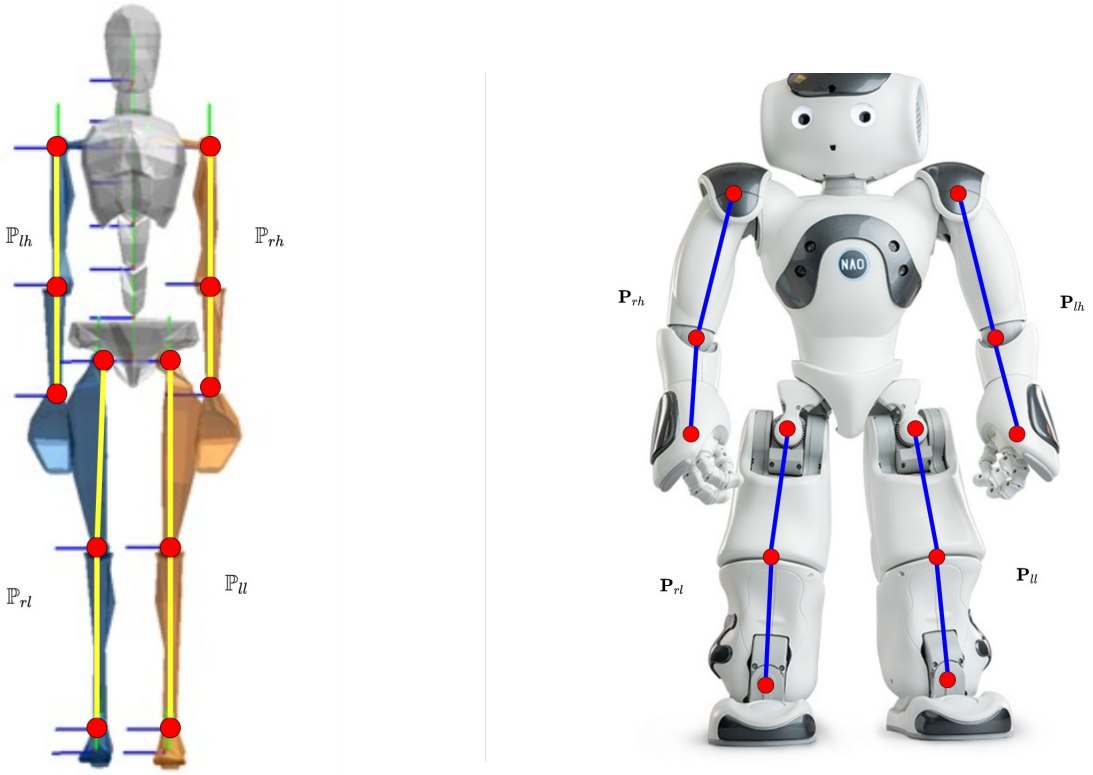

Figure 3.5: Human actor and NAO robot as M-DIP model

The individual pendulums states are comprised of roll and pitch angles in the robot frame and can be used the formulate the equation of motions from equation 2.9 [41]. The mass, velocity and acceleration contribution of the M-DIP as composition can be computed as,

Mass:

$$\mathbb{M} = m_{rl} + m_{ll} + m_{rh} + m_{rh} \tag{3.18}$$

Velocity:

$$\dot{\mathbb{X}}_i = m_i l_{sl} l_i \begin{bmatrix} 0 & -\sin(\gamma_{sl} - \gamma_i)\dot{\gamma}_i \\ \sin(\alpha_{sl} - \alpha_i)\dot{\alpha}_i & 0 \end{bmatrix} \quad , i = rh, rl, lh, ll \tag{3.19}$$

Acceleration:

$$\ddot{\mathbb{X}}_i = m_i l_{sl} l_i \begin{bmatrix} 0 & -\cos(\gamma_{sl} - \gamma_i) \\ \cos(\alpha_{sl} - \alpha_i) & 0 \end{bmatrix} \quad , i = rh, rl, lh, ll \tag{3.20}$$

where $i = sl$ represent the pendulum in contact with the environment (in this case, the legs); α_i and γ_i represent the roll and pitch angles of the respective pendulums i respectively. Given the velocity and acceleration contribution of the M-DIP to the composition model, the position of ZMP can also be retrieved as,

$$p_{ZMP,H} = \frac{1}{\mathbf{g}\mathbb{M}} \{ \ddot{\mathbb{X}}\ddot{\mathbb{P}} + \dot{\mathbb{X}}\dot{\mathbb{P}} + \mathbb{M} l_{sl} \mathbf{g} \begin{bmatrix} \sin(\gamma_{sl}) \\ -\sin(\gamma_{sl}) \end{bmatrix}_H \} \tag{3.21}$$

The equation 3.21 can also be rewritten for calculating the postion of robot ZMP as,

$$p_{ZMP,R} = \frac{1}{\mathbf{g}\mathbf{M}} \{ \ddot{\mathbf{X}}\ddot{\mathbf{P}} + \dot{\mathbf{X}}\dot{\mathbf{P}} + \mathbf{M} l_{sl} \mathbf{g} \begin{bmatrix} \sin(\gamma_{sl}) \\ -\sin(\gamma_{sl}) \end{bmatrix}_R \} \tag{3.22}$$

where,

$\mathbb{P}_i = [-\arctan(\frac{-y_i}{z_i}) \quad \arctan(\frac{-y_i}{z_i})]^T$ is the position state of the respective pendulum i;

$\dot{\mathbb{X}} = [\dot{\mathbb{X}}_i, i = rh, rl, lh, ll]^T$ is the velocity contribution of the pendulum model of human;

$\dot{\mathbf{X}} = [\dot{\mathbf{X}}_i, i = rh, rl, lh, ll]^T$ is the velocity controibution of the pendulum model of robot;

$\ddot{\mathbb{X}} = [\ddot{\mathbb{X}}_i, i = rh, rl, lh, ll]^T$ is the acceleration contribution of the pendulum of human;

$\ddot{\mathbf{X}} = [\ddot{\mathbf{X}}_i, i = rh, rl, lh, ll]^T$ is the acceleration contribution of the pendulum of robot.

Chapter 4

Hierrachical Quadratic Programming

In robotics, the use of inverse kinematics is very common and widely implemented for any applications. In humanoid robots, the robot structure is redundant which means it has more DoF than the number need to position the end-effector in the desired pose. These extra DOFs can be used to perform other tasks like *avoiding singularities, balancing, dodging obstacles, minimizing energy consumption, etc.*

Usually more than one aspect of the task must be transmitted to the robot. For this purpose, *task prioritization* will be used by allowing the jacobian of one task into the null projection of another task [39]. To implement the *task prioritization* effectively, an highly efficient method known as "Hierrachical Quadratic Programming (HQP)" can be used. HQP results in handling stacks of tasks with less computation intensity [42] . This chapter details the task specification and HQP for this research from the concepts explained in chapter 3.

4.1 Types of Tasks

A task or constraint refers to any geometric goals which must be achieved by the robot. For example, reaching a certain point in the world frame, reaching a particular joint angle, avoiding an obstacle, placing the center of mass in a specific location and so on. Another way of seeing them is as constraints which limit the robot's allowed motion: the robot can place itself in any way as long as it doesn't disturb the task.

Several tasks of each type will be performed. Each type will be described separately below. The mathematical description will be given as well as examples of when that type of task is useful. The way of arranging these tasks according to their priorities will be detailed. And

finally, the tasks used in this work will be defined.

4.1.1 Equality tasks

This type of task or constraint is the most commonly found when dealing with inverse kinematics. It consists of tracking values which can be described in function of the robot's joints. For example, the position and orientation of an end-effector, the position of the center of mass, and many other key points in the robot's body can be described in function of the joint configuration. For a given task involving DGM and IGM, let the pose descriptor be X and joint descriptor be q. Then the change in joint vector $\dot{q}$ with jacobian J is given by,

$$\dot{X} = J\dot{q} \tag{4.1}$$

Then the objective that need to be accomphlished can be represented in the task space velocity as $\dot{e}_H^*$. Once obtained, the robot input can be computed as a quadratic problem as,

$$Find \quad \dot{q}^* \in \min_{\dot{q}} ||J\dot{q} - \dot{e}_H^*|| \tag{4.2}$$

More generally, the linear constraints in equation 4.2 can be represented in the form $Ax = b$ as [42],

$$Find \quad x^* \in \min_{x} ||Ax - b|| \tag{4.3}$$

Among the possible task vector x^*, the pseudo inverse can be computed as ,

$$x = A^+ b$$

A^+ can be calculated from Singular Value Decomposition (SVD). This simplifies the calculation, since the pseudo-inverse of a diagonal matrix is found by taking the reciprocal of each of its non-zero elements [6].

$$A_L^+ = (A^t A)^{-1} A^t$$
$$A_R^+ = A^t (A A^t)^{-1} \tag{4.4}$$

From equation 4.4, the pseudo inverse can be calculated with less computation with linear independent columns or independent rows [6]. With these equality tasks, the cartesian tracking, CoM projection computation for balance, avoiding singularities and much mor can be achieved.

4.1.2 Inequality tasks

This type of tasks or constraints are most commonly used for bound type functions like joint limit, joint velocity limit and much more. It consists of maintaining values within the theshold limit Similar to equality-only least square quadratic program in equation 4.3, a set of linear inequality constraints can be written as,

$$Find \quad x^* \in \{x,\ s.t\ Cx \leq d\} \tag{4.5}$$

The above constraints are very useful, for example, for minimizing the diverence between captured human joint angles and the robot's angles. It is also commonly used in industrial robots to maximize the distance from the joint angles and the joint limits. Unlike equality tasks, optimization tasks cannot be hard constraints, since the optimal value might not be reached.

4.2 Quadratic Programming Approach

Humanoid robots are usually redundant with respective allowing to add more tasks to the task space. Ensuring task priorities for the assigned tasks will help the optimum performance. However, to achieve a task in three-dimensional space, will require infinite joint configurations to achieve a goal. The space where motions can be performed without perturbating the task is known as task's null projector. To efficiently compute and optimize the tasks for the application, hierrachical quadratic programming can be used as proposed in [24]. This optimization technique provides online faster computation and can cope up with the redundancy of the

humanoid robot.

4.2.1 Null Space Projection

The null space projector is directly linked to task Jacobian and is alternatively called as Jacobian's null projector. The act of performing a task within the other task is usually called as projecting the task vector Z_2 into the Jacobian J_1's null space. Then the projection on Jacobian J_1 is given by,

$$P_1 = I - A_1^+ A_1 \tag{4.6}$$

where I is the identity matrix. The task prioriy is set such that task j has higher prioirity than task $j+1$. Then equation 4.1 can be computed as quadratic problem as,

$$\min_x (\underline{A}_p x - \underline{b}_p)^T Q (\underline{A}_p x - \underline{b}_p) \tag{4.7}$$

such that,

$$\underline{A}_p = \begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_p \end{bmatrix} \qquad \underline{b}_p = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_p \end{bmatrix}$$

where $(\underline{A}_p, \underline{b}_p)$ is a set of non-conflicting linear constraints that need to be satisfied for best simultaneously and Q is the weighing matrix for optimizing the constraints for the tasks proposed. Then the second contraint (A_2, b_2) is solved with the null projector of task 1 with constraint (A_1, b_1) using the minimization of quadratic programming as [42],

$$\min_x ||A_2 P_1 x_2 - (b_2 - A_2 A_1^+ b_1)|| \tag{4.8}$$

Similarily the third constraint (A_3, b_3) on the null projection of task 2 can be given by,

$$x_2^* = (A_2 P_2)^+ (b_2 - A_2 A_1^+ b_1) + P_2 \tilde{x}_3 \tag{4.9}$$

and the complete solution solving (A_1, b_1) at its best and (A_2, b_2) if possible is given by,

$$x_2^* = A_1^+ b_1 + (A_2 P_2)^+ (b_2 - A_2 A_1^+ b_1) + P_2 \tilde{x}_3 \tag{4.10}$$

The solution in equation 4.10 can be extended recursively to solve the p levels of the hierrachy with $P_0 = I$, $x_0 = 0$ and $P_k = P_{k-1}\tilde{P}_k$ the projector into the null space of A_k is given by,

$$x_p^* = \Sigma_{k=1}^p (A_k P_{k-1})^+ (b_k - A_k A_{k-1}^+ b_{k-1}) + P_k \tilde{x}_{k+1} \tag{4.11}$$

The basic projector P in equation 4.11 can be replaced by the advanced projector suggested by Escande [43] as,

$$x_p^* = \Sigma_{k=1}^p Z_{k-1} (A_k Z_{k-1})^+ (b_k - A_k A_{k-1}^+ b_{k-1}) + Z_p z_{p+1} \tag{4.12}$$

where Z_k is a basis of a null space of $\underline{A}_k$ (*i.e.* $A_i Z_i = 0$ and $Z_i^T Z_i = I$, $i = 1, 2, ..., k$) and is a vector of dimension of null space of $\underline{A}_k$. The writing in last equation is more efficient than equation 4.11 due to its corresponding matrix size [42].

4.2.2 Hierrachy of Equality Quadratic Program

The Equality-only Hierrachical Quadratic Program is considered (eHQP) as a least-square problem. It is written as a set of p eHQP: at level k, the QP is solved using,

$$min_{x_k, w_k} ||w_k|| \tag{4.13}$$

$$s.t. \quad A_k x_k = b_k + w_k \tag{4.14}$$

$$\underline{A}_{k-1} x_k = \underline{b}_{k-1} + \underline{w}_{k-1}^* \tag{4.15}$$

where $\underline{A}_{k-1}, \underline{b}_{k-1}$ and $\underline{w}_{k-1}^*$ are the matrix and vectors composed of the stacked quantities as represented in equation 4.7. The weight $\underline{w}_{k-1}^*$ is a fixed value updated from the previous QP. The langrangian of the above equations give the optimality conditions for computing the weights $\underline{w}_k$ recursively [42].

$$w_k = A_k x_k - b_k \tag{4.16}$$

$$\underline{A}_{k-1} x_k = \underline{b}_{k-1} + \underline{w}^*_{k-1} \tag{4.17}$$

The hierrachy is kept such that $(k-1) \gg k \gg (k+1)$ to prioritize the planned constraints. Equation 4.16 confirms that higher level constraints are kept when solving for lower level tasks. The eHQP problem derived in this section allows equality constraints to be achieved at its best if possible. The tasks to be solved using eHQP are presented in section 4.3.

4.2.3 Hierrachy of Inequality Quadratic Program

As a minimization HQP problem, the linear inequality constraints in equation 4.5 for two tasks $(p = 2)$ is formulated as,

$$\min_{x, w_2} ||w_2|| \tag{4.18}$$

$$subject\ to\ C_1 x \leq d_1 + w_1^* \tag{4.19}$$

$$C_2 x \leq d_2 + w_2 \tag{4.20}$$

This iHQP problem can be solved using a classical active search. However, this would not take into account the specific form of the varibable w_2 In practice, its specific role in the constraint equations can be taken into account to reduce the amount of computation. Concluding the HQP problem containing both equality and inequality constraints, the formulation can be written as,

$$\dot{q}_i = \min_{\dot{q}, w_i} ||A_i \dot{q} - b_i||^2 + ||w||^2 \tag{4.21}$$

$$s.t.\ C_i \dot{q} - w \leq d_i \tag{4.22}$$

$$s.t.\ C_{i-1} \dot{q} \leq d_{i-1} \tag{4.23}$$

$$s.t.\ A_{i-1} \dot{q} = b_{i-1} \tag{4.24}$$

4.3 HQP in Motion Imitation

When imitating human motion, it is necessary to first clarify exactly which aspects of the motion are to be copied. Among these, it is also necessary to specify which ones are the most important. Is tracking the hand position in the world frame more important than tracking the elbow angle? Does the robot need to keep balance or is it attached to some structure? The answers to these questions will influence in the quantity, type and priority of the tasks chosen.

In this work, the following tasks will be considered: *joint-limit avoidance, end-effector tracking, joint tracking, ZMP balancing, posture tracking and fall-detection.* Each task will be discussed individually.

4.3.1 Task 1: Joint Limit Avoidance

To imitate the motion, staying away from joint limits should be a high priority of all. In fact, in any robotics applications, avoiding the joint limits is always the highest prioirity. The solution achieved from imitation should stay within the joint limit configurations and the resultant joint configuration shouldn't comply other tasks.

Staying away from the joint limits can be described as staying as close as possible to the median value for the joint position. The joint limit avoidance defines a bound on a function of the robot configuration:

$$e_{min}(\Omega) \leq e(q) \leq e_{max}(\Omega) \tag{4.25}$$

where q and Ω represent the current joint vector and the desired parameters of the robot motor limits. This linear inequality constraint is homogenous to the configuration and can be achieved by,

$$[e_{min}(\Omega) - e(q)] \leq J\dot{q} \leq [e_{max}(\Omega) - e(q)] \tag{4.26}$$

where J represent the robot Jacobian, a set of kinematic chains broken from the robot structure. Then the inequality constraints for the HQP be represented as,

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \dot{q} \leq \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{4.27}$$

In the above constraints, $(C_1, d_1) = ()$ and $(C_2, d_2) = ()$ are the constraint matrices and vectors of lower and upper joint limits respectively.

4.3.2 Task 2: Balance Control

The ZMP and CoM retargetting presented on section 3.2.1 can be formulated to obtain the optimal correction that satisfies the balance condition on the humanoid.

$$\min_{pZMP} (\dot{P}^{ref}_{CoM} - \dot{P}_{CoM})^T Q (\dot{P}^{ref}_{CoM} - \dot{P}_{CoM}) \tag{4.28}$$

$$such\ that\ \dot{P}_{CoM} = \dot{P}^*_{CoM} + \frac{1}{z^*_{CoM}} (P_{CoM} - P_{ZMP}) \tag{4.29}$$

$$lb_{SP} \leq P_{ZMP} \leq ub_{SP} \tag{4.30}$$

(lb_{SP}, ub_{SP}) is the lower and upper limit of Support Polygon; z^*_{CoM} is the height of the CoM in previous QP; Q is the weighing matrix The computation of CoM height using LIPM model helps in better estimation of ZMP position and consequently, a more accurate correction [3].

4.3.3 Task 3: End-effector tracking

To track the end-effector of the tree-structured robot, both position and orientations are considered. Considering the human actor task vector X_H and the robot task vector X_R, the end-effector tracking can be achieved and the change in joint vector $\dot{q}$ is,

$$\dot{q} = J^+ (X_H - X_R) \tag{4.31}$$

The equality constraints can then be computed using for the current joint configuration q^c,

$$A_3(q^c)\dot{q} = (X_H - X_R) = b_3 \tag{4.32}$$

where the constraint matrix $A_3(q^c) = J(q^c)$ is the robot jacobian; X_H and X_R are the set of end-effector states respective to human and robot.

4.3.4 Task 4: Joint trajectories tracking

To imitate the approximately exact motion from human actor data, the differene between joint vectors of human actor q_H and humanoid robot q_R need to be minimum resulting in inequality linear constraints and is represented as,

$$\min \kappa ||q_H - q_R||^2 \tag{4.33}$$

As a iHQP problem, equation 4.33 can be rewritten as ,

$$C_4 \dot{q} \leq d_4 \tag{4.34}$$

where $C_4 = I$ and $d_4 = \kappa(q_H - q_R)^2$ is the computation performed. κ is a custom weight matrix to provide compartively less weight to leg chains.

4.3.5 Task 4: Posture Control

The human actor and the robot are simplified to the composition of M-DIP model and the posture control can be implemented directly using the acceleration feedback between the two subjects. The acceleration control for the motion-retargetting can be implemented as a weighted-constraint based problem [44],

$$\min ||\ddot{\mathbb{X}}_H - \ddot{\mathbf{X}}_R|| \tag{4.35}$$

Chapter 5

Expermental Setup

5.1 Proposed Setup

This section describes the proposed setup for motion tracking suit and humanoid robot, NAO. Figure 5.1 represents the environment setup for this research. The first stage of this research was to implement the motion imitation with ZMP balance control. This implementation helps the initial validation of the capabilities for real-time motion imitaion, as well as processing the human data. In this case, the single support is initially tested since NAO robot will never lose it's balance when moving it's hands.

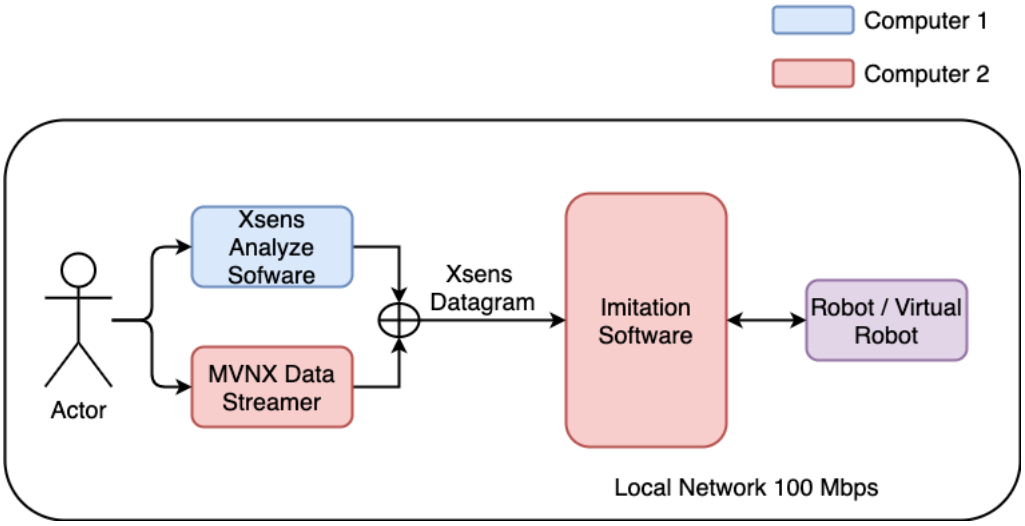

Figure 5.1: Illustration of the experimental setup

5.2 NAO robot platform

NAO is a programmable, autonomous, 57 cm tall humanoid robot developed by Aldebaran Robotics, a French startup company. The biped is equipped with several different sensors, including an inertial measurement unit with accelerometer, gyrometer and four ultrasonic sensors that provide NAO with stability and positioning within space, in addition eight force-sensing resistors and two bumpers. It features an Intel ATOM 1,6 GHz CPU, located in the head, that runs a Linux kernel and a second CPU located in the torso.

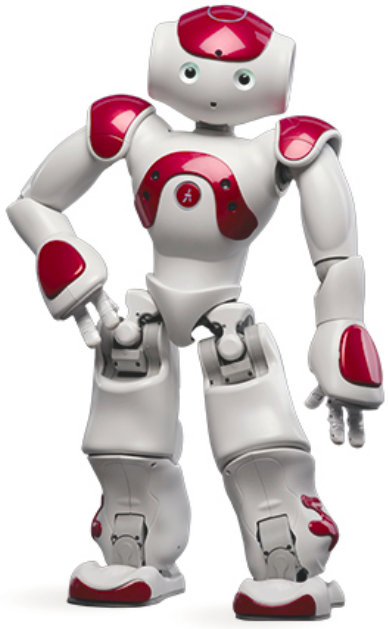

Figure 5.2: NAO - Aldebaran Robotics [5]

NAO has a total of 25 degrees of freedom (DOF), 11 DOF for the lower part that includes legs and pelvis, and 14 DOF for the upper part that includes trunk, arms and head. Each leg has 2 DOF at the ankle, 1 DOF at the knee and 2 DOF at the hip. A special mechanism composed of two coupled joints at each hip equips the pelvis. The rotation axis of these two joints are inclined at 45° towards the body. This mechanism replaces the classical set of three active rotary joints encountered in most humanoid robots.

Aldebaran Robotics provides a complete documentation for the robot, based on a geometric model shown in their website [5]. The model of the lower body of the robot NAO is obtained

considering the two legs identical and identically actuated. The two revolute joints that constitute the pelvis and named separately, but they essentially represent a unique actuated DOF. To conclude, it is important to remark that, unlike prototypes such as Wabian-2R or LOLA, the foot sole of the present version of NAO does not feature any passive or active joint that would enhance higher speed gait performances.

5.2.1 NAOqi C++ SDK

The robot's embedded software, *NAOqi*, a framework so the robot can be programmed in various operating systems, and in various languages, including C++, Python and MATLAB. The manufacturer also provides a software, *choreographe* allows connecting to the real robot through local network and starts the NAOqi service in the robot. *Choreographe* can also be used as a graphical interface for block-programming to interact with the robot.

In this research, the software is developed in C++ for interaction with the real robot and compiled with CMake, a cross compiled platform for build systems, to make use of NAOqi libraries. The support for NAO v5 humanoid framework is provided in NAOqi version 2.1.4 and is used for both real robot and simulation. The NAOqi 2.1.4 applications could be build in unix-systems by custom cross-compiler *qibuild*, a cmake extension and a python 2.7 module developed by Aldebaran Robotics. At the time of research, python 2.7 is completely abandoned. To make use of the functionalities of *qibuild*, the module is built from source on python 3.8.

The software is developed on unix-system and ran in a seperate windows computer communicating with the robot through local network. The software is developed as a standalone build for suitable version control using *git* and is available in [45].

5.2.2 CopelliaSim support

To simulate the dynamic behaviour during motion retargetting, the work is initially planned to simulate the robot in robotics simulators like Gazebo, Webots and CopelliaSim. The support for Webots and Gazebo has been dropped by Aldebaran community at the time of the research. Hence, the existing model of NAO robot in CopelliaSim is used to interact with NAOqi services and proxies.

Figure 5.3 presents the simulation of NAO robot and the interaction of Visual proxies from

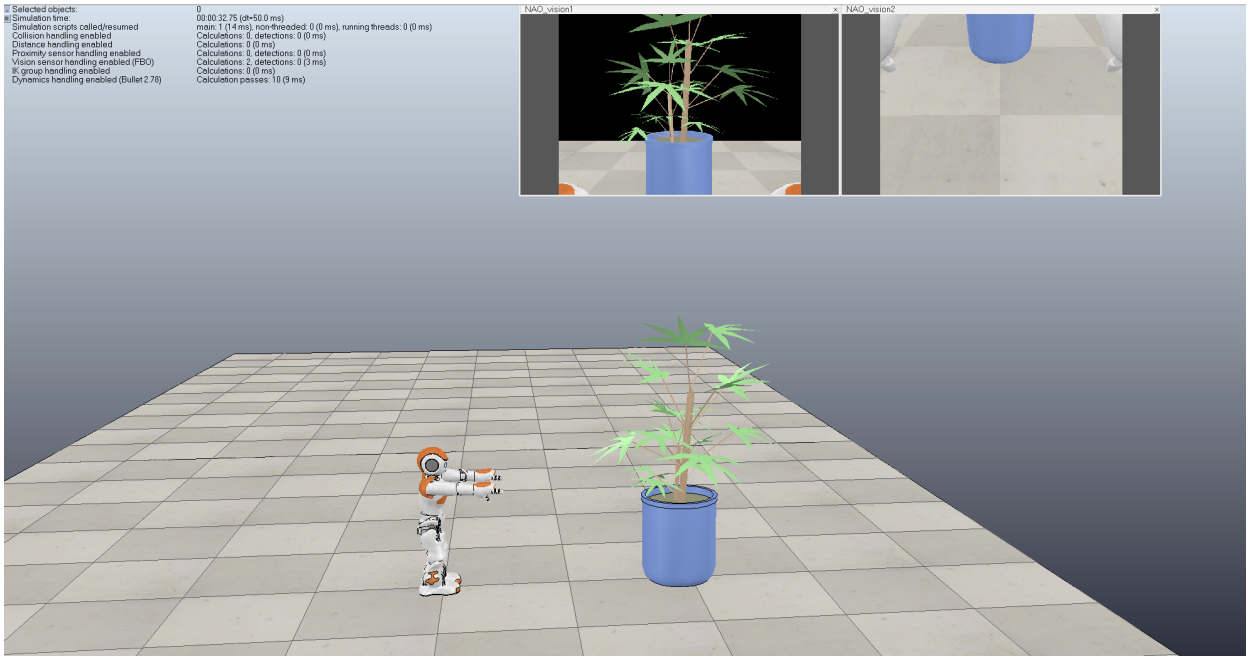

Figure 5.3: NAO robot in CopelliaSim simulator

NAOqi services. The simulation is used as a testing platform to monitor the initial behaviour during the imitation process, although there are evidences that the simulators doesn't replicate exactly the dynamic behaviour of the humanoid robot in the real world [2].

5.3 Xsens MVN Analyze

MVN Analyze/Animate, developed by Xsens, is a tool to capture and compute the 6DOF motion data of an inertial sensor-driven system. It allows the export of the data to third party applications such as Motion Builder, making the data available to drive rigged characters in e.g. animations. The data transfer to other applications is primarily file based when using MVN Analyze/Animate. With the XME API (SDK) there are many other options. Typically, Xsens inertial sensor technology is used for orientation, attitude- and positioning data. The technology is integrated by fleet operators, robotics integrator, digital mapping companies, drone developers, Vehicle-testing, Autonomous Vehicles, Warehouse and Logistics (AGV/AMR), Maritime/offshore, Aerospace, Heavy-Industry and Agriculture.In robotics research community, the MVN analyze system is used for motion imitation purposes due to its low-cost efficient sensor suit. This section describes the sensor suit setup and network streaming protocol over UDP networking.

Figure 5.4: Xsens MVN Analyze Suit

5.3.1 Sensor Setup

The hardware of Xsens MVN is available as a body-wired solution that streams data wirelessly to a PC/laptop (MVN Link) and a completely wireless solution(MVN Awinda). In the field of humanoid robotics and bio-mechanics, to capture the motion of the human body, 17 motion trackers are attached to the feet, lower legs, upper legs, pelvis, shoulders, sternum, head, upper arms, fore arms, and hands. The sensor modules are inertial and magnetic measurement units that contain 3D gyroscopes, 3D accelerometers and 3D magnetometers.

Each module comes with higher sensor fusion algorithms, with advnced signal processing frameworks like StrapDown Integration (SDI). Due to the use of SDI, 3D tracking accuracy of each motion tracker is equivalent both for MVN Link (240 Hz) as well as MVN Awinda (60 Hz), with only a reduced time resolution for the latter. Hence, for high dynamic movements including frequent interactions with the floor (contacts), MVN Link is recommended. Xsens MVN framework provides implementation in C++ and python. The Software Development Kit (SDK) allows fetching raw data for direct manipulation to our wishes. It can be noted that

the Xsens software has to setup to enable data sharing available outside the software.

5.3.2 Xsens Networking Protocol

The streaming feature in Xsens software enables the computer that run MVN Analyze to stream the captured data over a network to other client computers (in this case, the windows computer that runs imitation software). The network environment will be assumed to be a local 100 Mbit Ethernet network, larger network topologies are not considered and can be covered by file transfer of the already given file export functionality or later extensions to the network protocol. Thus, few packet loss or data corruption during transfer is to be expected, as well as constant connectivity.

Network communication uses a protocol stack, thus the streaming protocol will be implemented on top of a given set of protocols already available for the network clients. In this case, the layers to build upon are IP and UDP (or TCP, which is also supported). IP (Internet Protocol, RFC 791) is the network layer protocol used in Ethernet networks and defines the source and destination of the packets within the network. Upon this, UDP (User Datagram Protocol, RFC 768) is used to encapsulate the data. The UDP Protocol is unidirectional, and contrary to TCP (Transmission Control Protocol, RFC 793) it is stateless and does not require the receiver to answer incoming packets. This allows greater speed. The default port for xsens network streamer is 9763 and can be changed if needed.

5.3.3 Datagram Protocol

The motion capture data is sampled and sent at regular time intervals for which the length depends upon the configuration of MVN Analyze/Animate. Common sampling rates lie between 60 and 240 Hertz. The update rate of the real-time network stream can be modified separately. The data content in the datagram is defined by the specific protocol set, but basically, the positions and rotation of all segments of the body at a sampling instance are sent away as one or more UDP datagrams.

Each datagram starts with a 24-byte header followed by a variable number of bytes for each body segment, depending on the selected data protocol. All data is sent in 'network byte order', which corresponds to big-endian notation. The message format that are sent over the

network is detailed in the following subsections.

Message format

Each message streamed from Xsens software contains a header describing the type of the data and some identification information, so the receiving end can apply it to the right target.

6 bytes ID String
4 bytes sample counter
1 byte datagram counter
1 byte number of items
4 bytes time code
1 byte character ID
1 byte number of body segments – from MVN 2019
1 byte number of props – from MVN 2019
1 byte number of finger tracking data segments – from MVN 2019
2 bytes reserved for future use
2 bytes size of payload

Header ID string

The ID String is an ASCII string which consists of 6 characters (not terminated by a null character). It serves to unambiguously identify the UDP datagram as containing motion data of the format according to this specification. Since the values in the string are characters, this string is not converted to a big- endian notation, but the first byte is simply the first character, etc.

ASCII	M	X	T	P	x	y
Hex	4D	58	54	50	*hex(x)*	*hex(y)*

where xy be the desired message type that is being published. The required messages published for this research work is tabulated in Table 5.3.3. The message followed by the header string from above is the respective sensor data retrieved from the Xsens software. The messages are processed sequentially to the imitation software therby performing realtime imitaion.

xy	**Description**
02	**Pose data (Quaternion)**. Absolute position and orientation of sensor segments.
20	**Joint data**. Joint definition and angles
24	**Centre of Mass**. Absolute position of centre of mass

Table 5.1: Essential XSENS message type for Motion Retargetting

Data string

Once the header string is formed, the data string which collects the information of the message that needs to be published, processed and concatenated with the header string. These strings collectively should mostly be less than 1500 bytes and are sent through UDP Networks. The list of information that need to be processed are listed in order below.

- **Sample Counter -** The sample counter is a 32-bit unsigned integer value which is incremented by one, each time a new tracking data is processed.
- **Datagram Counter -** The size of the datagram is usually limited by Maximum Transmission Unit (MTU), until approx. 1500 bytes of the underlying network. If the data is split, then datagram counter helps in determining the number of data packages sent through single datagram
- **No. of items -**The number of items is stored in 8-bit unsigned integer value. The number indicates the number of segments or points being tracked.
- **Time Code -**The timecode represents the time data recorded by MVN Analyze in milliseconds. The timecode is stored in 32-bit unsigned integer value.
- **Character ID -** Since the research is focused on single user and will always be 0
- **Number of body segments -** The value contains the number of regular body segments of the character. In practice, this value is always 23.
- **Reserved bytes for future use -** Any left-over bytes near the end of the datagram header are reserved for future versions of this protocol.

- **Payload size -**The last 2 byte contain the size of the payload, meaning the size of the datagram without the header. This value can be ised when an unknown datagram is received to skip its contents in a reliable way.

Tracker data string

The tracker message at each timestep is then formatted as presented in table 5.3.3.

Data	**Datargam Format**
Pose (02)	4 bytes of segment ID 4 bytes x-coordinate of segment position 4 bytes y-coordinate of segment position 4 bytes z-coordinate of segment position 4 bytes x rotation of segment rotation 4 bytes y rotation of segment rotation 4 bytes z rotation of segment rotation
Joint Angles (20)	4 bytes point ID of parent segment connection 4 bytes point ID of child segment connection 4 bytes floating point rotation around segment x-axis 4 bytes floating point rotation around segment y-axis 4 bytes floating point rotation around segment z-axis
Centre of Mass (24)	4 bytes x-coordinates of CoM position 4 bytes y-coordinates of CoM position 4 bytes z-coordinates of CoM position

Table 5.2: Tracking Data Message Configuration

Once each part of the datagram string is formatted, the string message is concatenated as

$$\textbf{Message} = \textbf{HeaderString} + \textbf{DataString} + \textbf{TrackerDataString}$$

The message is published within 100 Mbps local network either through Ethernet or WiFi. The message parser and formatter has been developed in Python3 due to its ease functionalities with formatting XML files.

5.3.4 MVN Data Streamer

Since only a set of actions from motion capture suit are recorded and validated on the real robot, the files are exported to *MVNX* format, a prioritized XML format provided by Xsens. A custom *python* scripts are developed for procssing MVNX files to the desired format. A sample of data plotter developed is preseted in Figure 5.5. The data once captured either

directly from the software or from MVNX files are further processed and sent to the real robot for imitation.

Figure 5.5: Xsens data plot pelvis joint

The figure 5.5 represent the position and CoM plot of pelvis joint during the recorded motion. The initial stage of the motion is clearly upper body imitation followed by single support and movement along frontal plane of the human actor.

Chapter 6

Implementation

This chapter primarily discusses the steps and modifications that has been carried out to tune the software. The imitation is performed initially with upper body and then to lower body eventually testing the complex motions considering the NAO robot's structure. The results of the implementation will be discussed in the coming sections later in this chapter.

6.1 Methodology

The implementation began by testing the robot's kinematics already implemented in order to obtain relation between it's joints and task spaces. Once the suitable implementation has been validated, the captured human data is scaled down to the robot's size and it's desired parameters Ω are retrieved for HQP.

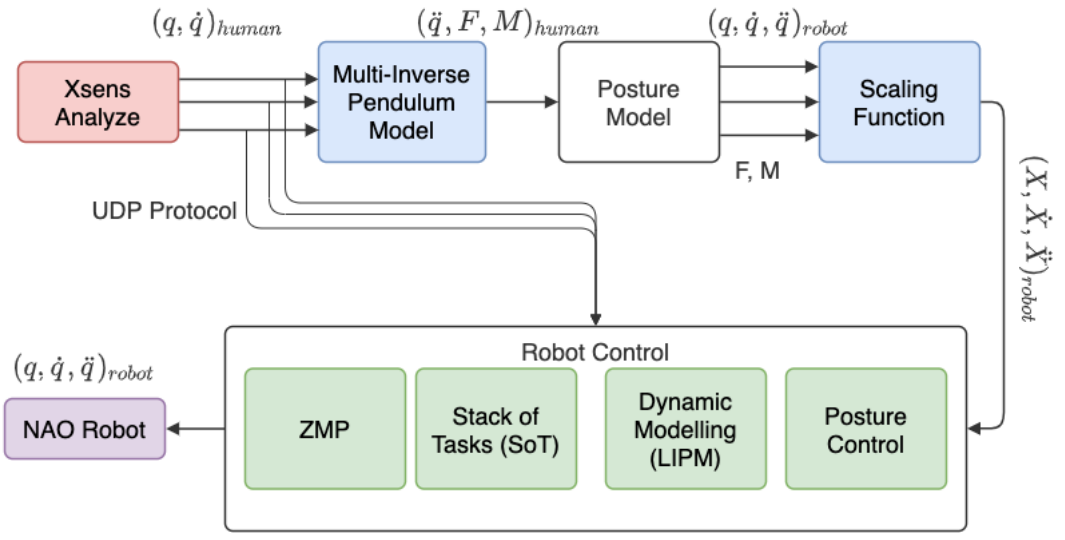

Figure 6.1: Block Diagram Representation of Imitation Program

Figure 6.1 represents the flow of data between individual blocks discussed in chapter 3 and chapter 4. It is notable that two computers are used for this implementation to extract convenient results.

6.2 Robot Model

During the course of this research, both real robot and virtual models are used. The real robot used for validation is a NAO v5 H25 Humanoid robot and the virtual model is a CopelliaSim model with approximately equivalent inertial modelling. Like previous implementations on kinematic modelling [6], the hand operations (open/close) will be ignored therby controlling 23 DoFs in total (vector q). Figure 6.2 represents the arrangement of DoFs that will be used in this research.

$$q = \begin{bmatrix} q_1 & q_2 & \dots & q_{23} \end{bmatrix}^T$$

6.2.1 Kinematic Modelling

The robot's geometric model was calculated using the nominal approach with *Modified Denavit-Hartenberg* (MDH) parameters. Since the robot is a tree structure, some links have more than one child links. In addition, it might be convenient to include frames in the model which are not necessarily defined according to the previous rules. Then, the kinematic model can be represented as,

$$\begin{bmatrix} \dot{x}_{lhand} \\ \dot{x}_{rhand} \\ \dot{x}_{xleg} \end{bmatrix} = J \begin{bmatrix} \dot{q}_{lhand} \\ \dot{q}_{rhand} \\ \dot{q}_{xleg} \end{bmatrix} \tag{6.1}$$

Equation 6.1 can be simply represented as $\dot{X} = J\dot{q}$ in the next sections where J is the robot Jacobian and x_i is a 3×1 vector without considering the end-effector oriendation. $xleg$ refers to $lleg$ and $rleg$ depending on the support phase of the robot. Considering both feet of the robot for kinematic modelling uses all the joints q, the robot will not be redundant and adding

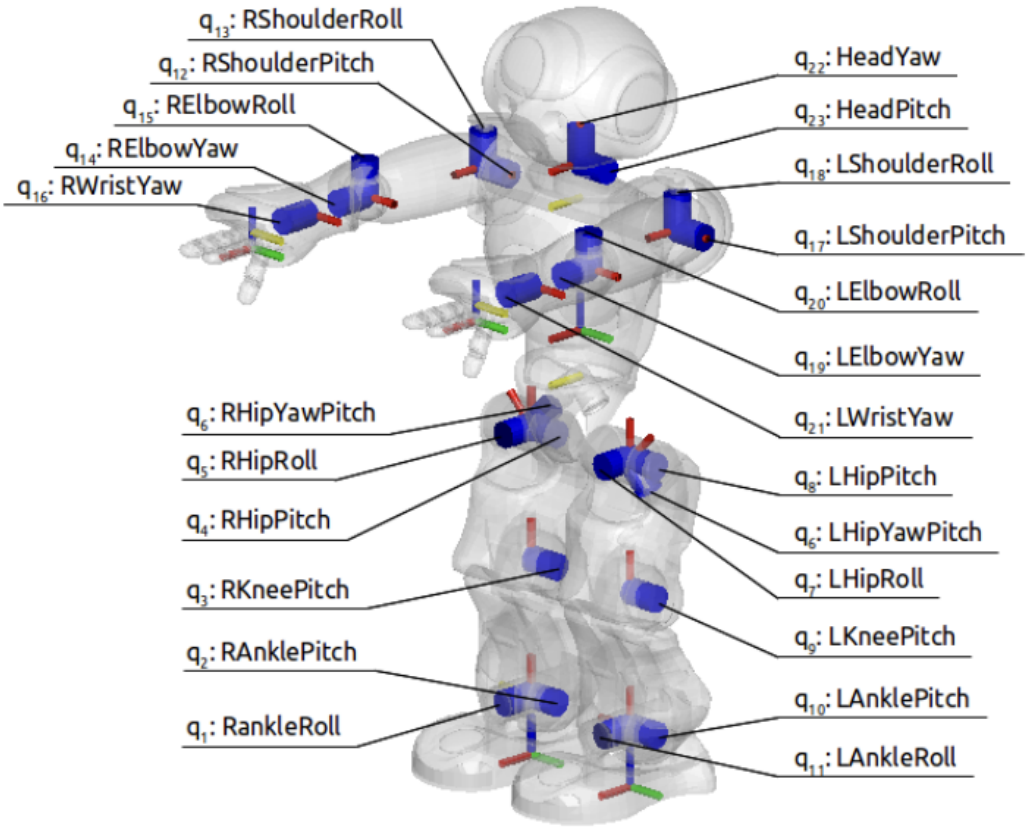

Figure 6.2: NAO Robot's DoF Arrangement [6]

tasks will cause overconstraint. To avoid the situation, two seperate models were created such that the above equation has a vector size of $(6 + 5 + 5) \times 1$ considering only three kinematic chains for the sake of redundancy. In other words, either left foot or right foot will always be in contact with the environment (*i.e.* floor). This work was implemented in MATLAB and was exported to a C++ library "Jacobian" (available in [45]).

6.2.2 Support Phases

Since the kinematic analysis is composed of two models for each single support (right foot and left foot), the computation is extended to a lighter version compared to the computation of 23 joints. The transition between single support (SS) and double support (DS) is tabulated in Table 6.2.2 which introduces the constraints of using the kinematic models developed in the above section.

In table 6.2.2, FH refers to the feet height of NAO, LH refers to the lift height during single support of the respective legs. $X_{balance}$ refers to the optimal projection of CoM onto the

horizontal plane during the imitation process.

Support	Pos. of abs. frame	X	$X_{balance}$	DS (constraint)	SS (constraint)
Right Foot	Right Foot	$\begin{bmatrix} x_{lhand} \\ x_{rhand} \\ x_{lankle} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$z_{lankle} = FH$	$z_{lankle} = FH + LH$
Both feet	Right Foot	$\begin{bmatrix} x_{lhand} \\ x_{rhand} \\ x_{lankle} \\ z_{lfoot} = 0 \\ z_{ltoe} = 0 \end{bmatrix}$	$\begin{bmatrix} \frac{x_{lfoot}}{2} \\ \frac{y_{lfoot}}{2} \end{bmatrix}$	-	-
Left Foot	Left Foot	$\begin{bmatrix} x_{lhand} \\ x_{rhand} \\ x_{rankle} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$z_{rankle} = FH$	$z_{rankle} = FH + LH$

Table 6.1: Transition between support phases

6.2.3 Masses and CoM

The robot's dynamic parameters like masses and Centre of Masses can be obtained from the documentation directly [5]. The individual masses and CoM for each link is calculated and documented in the website. To calculate the single point mass and the projection of CoM onto xy plane is,

$$M = \Sigma_{i=1}^{n} m_i \tag{6.2}$$

$$P_{CoM} = \frac{1}{M} \Sigma_{i=1}^{n} m_i c_i \tag{6.3}$$

where m_i and c_i represent the individual mass and CoM of link i obtained from documentation respectively; n is the number of moving masses in the robot body. The jacobian of CoM can then be formulated with respect to the robot frame (frame located at right foot) is given by,

$$J_{CoM} = \frac{\partial P_{CoM}}{\partial q} \quad or \tag{6.4}$$

$$\dot{P}_{CoM} = J_{CoM} \dot{q} \tag{6.5}$$

The size of J_{CoM} is 2×23 (reflection of CoM onto 2D plane (x, y)). P_{CoM} is a 2-dimensional vector since the projection on xy plane.

6.3 Human Motion

The sensor data acquired from Xsens suit is high-quality less-noise data with the ability to record wide range of motions. The data acquired greatly differ in cartesian sizing compared to the size of the NAO robot. Usually in motion imitation, the robot must reach the same point in task space as the human actor, but due to greater difference in size, the task space is treated as relative instead of absolute. To achieve this criteria, a scaling function need to be implemented before feeding the Xsens data to the robot control.

Scaling Function

In a previous work [46], the use of a scaling factor for each limb based on the ratio between the lengths of the kinematic chain of actor and imitator was proposed. This factor was then multiplied by the captured human data to obtain the desired positions for the robot. Since each limb was a simple serial chain, a proportional scaling factor could be applied satisfactorily.

However, in the present work, the balance is being taken into account and thus the robot has to be modelled as a whole, which gives rise to s tree-structure. In this case, keeping the direction of the captured human data means that the robot's end-effectors will fall on the line which connects the tracked points to the origin on the right foot.

A new scaling method is proposed in [36] which takes into account not only the lengths of each of the segments but also each of their directions. This way, the captured human motion is scaled segment by segment to the lengths of the robot while keeping the direction of that segment. The iterative process starts from the right ankle and moves upwards towards each of the end-effectors.

In summary considering a link l, the scaling can be processed as,

$$P_l^{robot} = \frac{P_l^{human} - P_a^{human}}{||P_l^{human} - P_a^{human}||} l_l^{robot} + P_a^{robot} \tag{6.6}$$

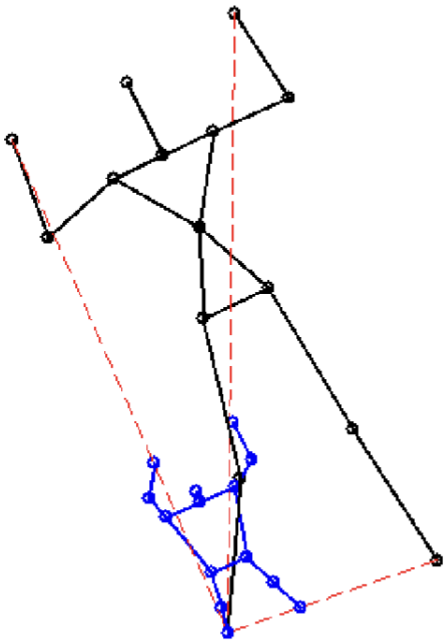

Figure 6.3: Proportional scaling of human motion [6]

where P_a refers to the antecedent position to the link l. Since the robot doesn't have a spine, the distances connecting shoulders and hips should remain constant. The shoulders cannot be scaled directly from the hips because of the different torso proportions between the human and the robot. It was chosen to scale a segment which goes from the point between the hips to the point between the shoulders, preserving symmetry. Also the position of absolute frame during single support is at right foot at any case. Considering the support changes into imitation, the absolute frame cannot stay at right frame and hence moved to the center of the feet. A threshold of 200 mm is set to avoid false positives due to jittering.

6.4 Robot Control

For the robot control part, the program is developed with the possiblity of extending as an application to the NAO community. The data received from Xsens is processed and sent to the robot control for motion imitation. A flowchart of the program is shown on Figure 6.4.

Each block of the flowchart is individually explained in Chapter 3 and Chapter 4. However an summary of each block is provided in subsections below.

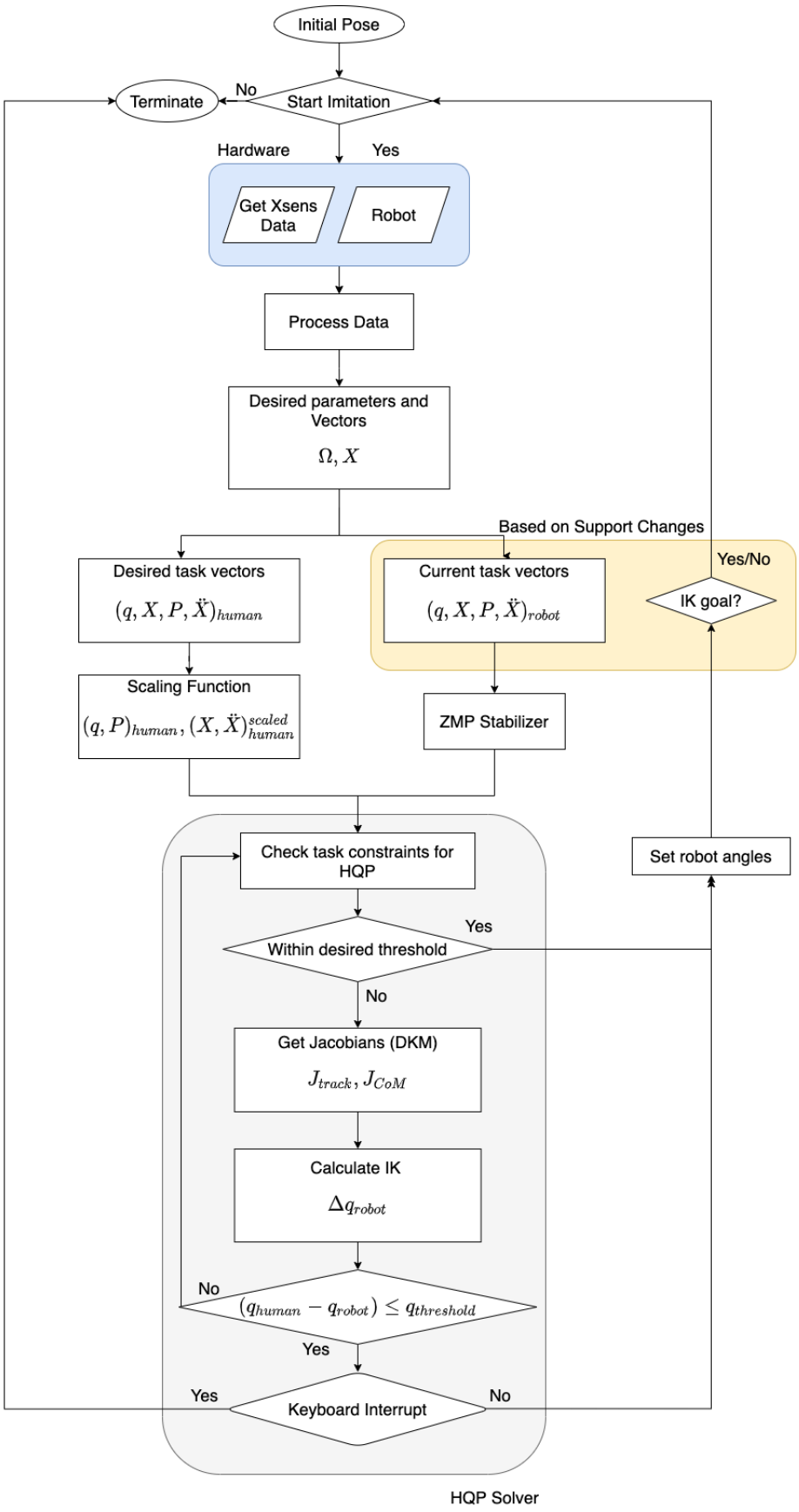

Figure 6.4: Flowchart Representation of Imitation Program

Initial Pose

The initial poses are the identity poses used when calibrating Xsens suit. In this work, only N-pose will be considered as the initial pose in both the case of calibration and the experimentation. Once the experimentation starts, a delay of 5 seconds will let the xsens user to pose in N-pose followed by the imitation program.

Start Imitation

Start Imitation allows the user to define the configurations for the imitation program. An additional section for letting the user to use recorded motion or real-time imitation can also be setup depending on the situation. If the imitation is online, the program will wait for the first message to be delivered by Xsens Analyze software.

Hardware

The hardware interface lets the xsens and NAO services get started to check connectivity between services and joint data buffers. Once the data is acquired without any interference the actual imitation is started.

Process data

The acquired UDP message from Xsens needs to be processed to a readable numerical format for further actions. The NAOqi proxy lets collecting data from robot joint buffers and memory buffers. The real-time sensors and encoders can be acquired from ALProxy, ALMotion and ALMemory services.

Desired Task Vectors

The desired task vectors are processed from human actor. The essential parameters like $X, \ddot{X}, P_{CoM,H}$ and q_H are assigned to the temporary memory buffer and scaled to the size of the robot. The vectors and matrices are formatted to the required dimensions of HQP solver.

Current Task Vectors

The current task vectors are derived from robot memory buffer. The memory buffer is handled using ALMemory proxy and ALPose proxy and are formatted to the vector and matrices.

HQP Solver

The task vectors and constraints described in section 4.3 will be programmed in C++ using the library *QPOases*.

Several iterations might be needed for the IK to reach the final goal, even if the final goal has been scaled down. However, since this is a numerical approach, it is not assured that a solution will be found. In fact, after some iterations, if the hard constraints are not reachable or compatible, the pseudo-inverse won't come to any useful results. Thus, a maximum number of iterations is set beforehand, so if the IK is taking too many iter- ations to be solved, the current goal can be abandoned and new human data should be taken. It was observed that good results are usually obtained with less than 6 iterations, so a safe limit was set at maxIterations = 20 [6].

6.5 Results

The

Conclusion

Humanoid robots are redundant open serial chains not fixed to their environment with a few dozen DOF. In order to cope with this complexity, the approach of imitation uses captured human motion as the basis for motion generation. This approach takes much of the burden of motion prediction away from the robot.

Due to the kinematic and dynamic differences between human beings and humanoid robots, motion transfer is not a straightforward matter. Besides taking into account the physical limitations of the robot, issues such as kinematic singularities, self-collision and balance must be addressed. Moreover, for real-time imitation, time constraints must be enforced.

In this research, whole-body motion imitation of human motion by the humanoid robot NAO using hierrachical quadratic programming was implemented with five tasks: tracking end-effectors, keeping static balance, avoiding joint limits, posture control and tracking human joint values.

A method to scale the human motion to the robot's dimensions one body segment at a time was introduced. The scaled end-effector positions in Cartesian space were tracked while keeping static balance for single support on either foot, double support and transition between supports. The joint limits were avoided by a combination of a clamping loop and a task which minimizes the distance from the average joint value. The joint values were kept as close as possible to the performer's whenever that didn't affect the higher priority tasks to imitate the manner as well as possible. More information and discussion will be added on the final version of the report.

Bibliography

[1] Y. Yang, V. Ivan, W. Merkt, and S. Vijayakumar, "Scaling sampling-based motion planning to humanoid robots," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2016, pp. 1448–1454.

[2] O. E. Ramos Ponce, "Generation of the whole-body motion for humanoid robots with the complete dynamics," Theses, Universite Toulouse III Paul Sabatier, Oct. 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01134313

[3] L. Penco, B. Clément, V. Modugno, E. M. Hoffman, G. Nava, D. Pucci, N. Tsagarakis, J.-B. Mouret, and S. Ivaldi, "Robust Real-time Whole-Body Motion Retargeting from Human to Humanoid," in *HUMANOIDS 2018 - IEEE-RAS 18th International Conference on Humanoid Robots*, Beijing, China, Nov. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01895145

[4] ——, "Robust Real-time Whole-Body Motion Retargeting from Human to Humanoid," in *HUMANOIDS 2018 - IEEE-RAS 18th International Conference on Humanoid Robots*, Beijing, China, Nov. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01895145

[5] N. A. Robotics. Masses, centre of masses and inertial matrix for nao v5 h25. [Online]. Available: http://doc.aldebaran.com/2-1/family/robots/masses_robot.html

[6] L. Poubel, "Whole-body online human motion imitation by a humanoid robot using task specification," 2013.

[7] N. Fukaya, S. Toyama, T. Asfour, and R. Dillmann, "Design of a humanoid hand for human friendly robotics applications," in *Human Friendly Mechatronics*, E. Arai, T. Arai,

and M. Takano, Eds. Amsterdam: Elsevier Science, 2001, pp. 273 – 278. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780444506498500462

[8] C. C. Kemp, P. Fitzpatrick, H. Hirukawa, K. Yokoi, K. Harada, and Y. Matsumoto, *Humanoids*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1307–1333. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_57

[9] M. Akhtaruzzaman and A. Shafie, "Evolution of humanoid robot and contribution of various countries in advancing the research and development of the platform," 11 2010, pp. 1021 – 1028.

[10] K. Tanie, "Humanoid robot and its application possibility," in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI2003.*, 2003, pp. 213–214.

[11] B. Rosenhahn, R. Klette, D. Metaxas, and D. N. Metaxas, *Human Motion Understanding, Modelling, Capture, and Animation / edited by Bodo Rosenhahn, Reinhard Klette, Dimitris Metaxas.*, 1st ed., ser. Computational Imaging and Vision, 36. Dordrecht: Springer Netherlands : Imprint: Springer, 2008.

[12] M. Eadweard, *Muybridge's complete human and animal locomotion : all 787 plates from the 1887 animal locomotion / by Eadweard Muybridge ; introd. to the Dover ed. by Anita Ventura Mozley,...*, ser. Collections of fine art in Dover books. New York: Dover.

[13] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991. [Online]. Available: https://doi.org/10.1177/027836499101000604

[14] S. LaValle, "Planning algorithms," 2006.

[15] Y. Zhang, J. Luo, K. Hauser, H. A. Park, M. Paldhe, C. G. Lee, R. Ellenberg, B. Killen, P. Oh, J. H. Oh *et al.*, "Motion planning and control of ladder climbing on drc-hubo for darpa robotics challenge," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2086–2086.

[16] E. Yoshida, C. Esteves, I. Belousov, J.-P. Laumond, T. Sakaguchi, and K. Yokoi, "Planning 3-d collision-free dynamic robotic motion through iterative reshaping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1186–1198, 2008.

[17] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[18] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[19] Y. Nakamura, *Advanced robotics: redundancy and optimization.* Addison-Wesley Longman Publishing Co., Inc., 1990.

[20] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* IEEE, 2005, pp. 238–244.

[21] E. Yoshida, O. Kanoun, C. Esteves, and J.-P. Laumond, "Task-driven support polygon reshaping for humanoids," in *2006 6th IEEE-RAS International Conference on Humanoid Robots.* IEEE, 2006, pp. 208–213.

[22] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 60–72, 2007.

[23] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond, "Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots," in *2009 IEEE international conference on robotics and automation.* IEEE, 2009, pp. 2939–2944.

[24] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[25] G. Jarquín, A. Escande, G. Arechavaleta, T. Moulard, E. Yoshida, and V. Parra-Vega, "Real-time smooth task transitions for hierarchical inverse kinematics," in *2013 13th IEEE-*

RAS International Conference on Humanoid Robots (Humanoids). IEEE, 2013, pp. 528–533.

[26] R. Featherstone and D. Orin, "Robot dynamics: equations and algorithms," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 826–834.

[27] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[28] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 29–43, 2004.

[29] L. Sentis and O. Khatib, "Control of free-floating humanoid robots through task prioritization," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 1718–1723.

[30] M. Popovic, A. Hofmann, and H. Herr, "Angular momentum regulation during human walking: biomechanics and control," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3. IEEE, 2004, pp. 2405–2411.

[31] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[32] A. I. Elhasairi, *Humanoid robot full-body control & balance restoration*. University of Surrey (United Kingdom), 2015.

[33] C. G. Atkeson and B. Stephens, "Multiple balance strategies from one optimization criterion," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2007, pp. 57–64.

[34] D. Katić and M. Vukobratović, "Survey of intelligent control techniques for humanoid robots," *Journal of Intelligent and Robotic Systems*, vol. 37, no. 2, pp. 117–141, 2003.

[35] S. S. Karthick Munirathinam, "Different approaches in dynamic motion imitation from human being to a humanoid robot," Theses, Ecole Centrale de Nantes, 2016.

[36] S. Sakka, L. Penna Poubel, and D. Cehajic, "Tasks prioritization for whole-body realtime imitation of human motion by humanoid robots," in *Digital Intelligence (DI2014)*, Nantes, France, Sep. 2014, p. 5 p. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01054887

[37] S. Mukherjee, D. Paramkusam, and S. K. Dwivedy, "Inverse kinematics of a nao humanoid robot using kinect to track and imitate human motion," in *2015 International Conference on Robotics, Automation, Control and Embedded Systems (RACE)*. IEEE, 2015, pp. 1–7.

[38] D. F. Brown, A. Macchietto, K. Yin, and V. Zordan, "Control of rotational dynamics for ground behaviors," in *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2013, pp. 55–61.

[39] W. Khalil and E. Dombre, *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

[40] W. Khalil, "Unified dynamic modeling of robots using newton-euler techniques."

[41] P. Pierro, "Stabilizer architecture for humanoid robots collaborating with humans," 2012.

[42] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[43] ——, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3733–3738.

[44] C. Van Loan, "On the method of weighting for equality-constrained least-squares problems," *SIAM Journal on Numerical Analysis*, vol. 22, no. 5, pp. 851–864, 1985.

[45] H. S. Selvakumar. Nao dynamics motion retargetting - nao-dynamics-imitation. [Online]. Available: https://github.com/franklinselva/nao-dynamics-imitation

[46] D. Cehajic, "Using human motion to control the humanoid robot nao." Ecole Centrale de Nantes, 2012.