Human-humanoid collaborative object transportation

Don Joven Agravante

Collège Doctoral du Languedoc-Roussillon

THÈSE

Pour obtenir le grade de

Docteur

Délivré par l'**Université de Montpellier**

Préparée au sein de l'école doctorale
I2S – Information, Structures, Systèmes

Et de l'unité de recherche
CNRS-UM LIRMM, IDH, UMR 5506

Spécialité:
SYAM – Systèmes Automatiques et Microélectroniques

Présentée par **Don Joven Agravante**
donjoven.agravante@lirmm.fr

Human-humanoid collaborative object transportation

Soutenue le 16/12/2015 devant le jury composé de

François Chaumette	DR	INRIA, Irisa Rennes	Président
Gabriel Abba	PU	Arts et Métiers ParisTech	Rapporteur
Florent Lamiraux	DR	LAAS-CNRS	Rapporteur
Pierre-Brice Wieber	CR	INRIA, Rhone-Alpes	Examinateur
Andrea Cherubini	MCF	UM-LIRMM	Co-encadrant
Abderrahmane Kheddar	DR	CNRS-AIST	Directeur de thèse

Université Montpellier

Contents

Acknowledgments

I would like to thank my thesis advisers Andrea Cherubini and Abderrahmane Kheddar for the guidance and support all throughout this endeavor.

I would also like to thank Gabriel Abba and Florent Lamiraux for taking the time to review this manuscript and to François Chaumette and Pierre-Brice Wieber for examining this thesis. All of your comments, inquiries and suggestions especially during my defense were very much appreciated.

I would also like to express my deep gratitude to the people who have given a tremendous effort in helping me with the various portions of this thesis. To Antoine Bussy for the time and effort to get me started in the field of physical human-robot interaction, to Alexander Sherikov and Pierre-Brice Wieber for collaborating in the conceptualization and development of the walking pattern generators, to Joris Vaillant for sharing his excellent code base on whole-body control and the theoretical and practical discussions on this matter. To all the people who have contributed to this work in some way or another, I am truly indebted to you.

Throughout this undertaking, I had met a lot of great people who shared common experiences, interests, hobbies and humor. To everyone, especially Joris, Alejandro, Ben C., François K., Romain, Antoine, Kevin, Vincent, Ben N., Adrien E., Adrien P., Alfonso, Antonio, Alexander, Damien, Pierre G., Stan, Airi, Tu-Hoa, Gerardo, Herve, Thomas M., Julie, Karim, David, Helene, Marion, Mariam, Thomas G., Celine, Matthias and Mael . Thank you.

To my family and friends from home, I am forever grateful for your unwavering support.

List of Symbols

$\mathbb{R}$ is the set of real numbers.

DOF is an acronym for Degrees Of Freedom of a robot.

CoM is an acronym for Center of Mass.

MPC is an acronym for Model Predictive Control.

ZMP is an acronym for Zero Moment Point.

$\mathbf{q} \in \mathbb{R}^{\mathrm{n}}$ is a vector of joint positions for an n-DOF robot.

$\boldsymbol{\tau} \in \mathbb{R}^{\mathrm{n}}$ is a vector of joint torques for an n-DOF robot.

$\mathbf{J} \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$ is a Jacobian matrix for an n-DOF robot where m is the number of task variables, the *classical* robot Jacobian has $m = 6$ with the task defined in 3D Cartesian space.

$\mathbf{C} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ is a matrix that relates joint velocities to the vector of Coriolis and centrifugal terms for an n-DOF robot.

$\mathbf{H} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ is a symmetric, positive-definite joint-space inertia matrix for an n-DOF robot.

$\boldsymbol{\tau}_{\mathbf{g}} \in \mathbb{R}^{\mathrm{n}}$ is a vector of torques due to gravity for an n-DOF robot.

$\mathbf{e} \in \mathbb{R}^{\mathrm{m}}$ is a vector of m task variables.

x, y, z as superscripts are used to denote the 3 dimensions in Euclidean space.

$\mathbf{t} \in \mathbb{R}^3$ is a translation/position vector of an object in a reference frame, for example ${}^{\mathrm{a}}\mathbf{t}_{\mathrm{b}} = [t^{\mathrm{x}}\ t^{\mathrm{y}}\ t^{\mathrm{z}}]^\top$ denotes the translation of b in the a reference frame.

$\boldsymbol{\theta} \in \mathbb{R}^{\mathrm{n}}$ is an orientation vector of an object in a reference frame where n denotes the number variables to express an orientation. Examples include Euler angles and quaternions.

$\mathbf{R}$ is a rotation matrix of an object in a reference frame, for example ${}^{\mathrm{a}}\mathbf{R}_{\mathrm{b}}$ denotes the rotation of b in the a reference frame.

$\mathbf{p} \in \mathbb{R}^{(3+\mathrm{n})}$ is a generalized pose vector containing both translation and rotation information where n denotes the number variables to express an orientation.

$\mathbf{T}$ is a generalized transformation matrix which can be used to transform a pose, motion or wrench to another reference frame, for example ${}^{\mathrm{a}}\mathbf{T}_{\mathrm{b}}$ denotes the transformation of b to the a reference frame. It contains both the rotation and translation (pose) information structured according to the variable being transformed.

$\mathbf{f} \in \mathbb{R}^3$ is a force vector such that $\mathbf{f} = [f^{\mathrm{x}}\ f^{\mathrm{y}}\ f^{\mathrm{z}}]^\top$.

$\mathbf{n} \in \mathbb{R}^3$ is a torque vector such that $\mathbf{n} = [n^{\mathrm{x}}\ n^{\mathrm{y}}\ n^{\mathrm{z}}]^\top$.

$\mathbf{h} \in \mathbb{R}^6$ is a wrench vector such that $\mathbf{h} = [\mathbf{f}^\top\ \mathbf{n}^\top]^\top$.

$\mathbf{L} \in \mathbb{R}^3$ is a vector of the angular momentum such that $\mathbf{L} = [L^{\mathrm{x}}\ L^{\mathrm{y}}\ L^{\mathrm{z}}]^\top$.

$\mathbf{G}$ is a gain matrix of appropriate dimensions.

m is the mass of an object or virtual model.

b is the damping of an object or virtual model.

k is the stiffness of an object or virtual model.

$\mathbf{x} \in \mathbb{R}^{\mathrm{n}}$ is a vector of the independent variables to optimize in a mathematical optimization problem where n is the number of variables.

$f(\mathbf{x})$: $\mathbb{R}^{\mathrm{n}} \to \mathbb{R}$ is an objective function to minimize or maximize in an optimization problem.

$\mathbf{Q}_{\mathrm{qp}} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ is a matrix used to define a standard quadratic program's objective function where n is the number of variables.

$\mathbf{c}_{\mathrm{qp}} \in \mathbb{R}^{\mathrm{n}}$ is a vector used to define a standard quadratic program's objective function where n is the number of variables.

w is a weighting parameter to control the relative importance between several objective functions in a mathematical optimization problem.

$\boldsymbol{\lambda} \in \mathbb{R}^{\mathrm{nm}}$ is a vector representing the linearized friction cone base weights where n is the number of contact forces and m is the number of generators chosen to represent the linearization.

$\mathbf{K}_{\mathbf{fc}} \in \mathbb{R}^{3\mathrm{n}\times\mathrm{nm}}$ is a matrix of generators for linearizing the friction cone where n is the number of contact points and m is the number of generators chosen to represent the linearization such that a concatenated vector of contact forces $\mathbf{f} = \mathbf{K}_{\mathrm{fc}}\boldsymbol{\lambda}$.

$\underline{n}$ signifies the lower limit of a constraint on variable n in an optimization problem.

$\overline{n}$ signifies the upper limit of a constraint on variable n in an optimization problem.

$\mathbf{c} \in \mathbb{R}^3$ is the position of the Center of Mass (CoM), it is shorthand for $\mathbf{t}_{\text{CoM}}$.

$\mathbf{z} \in \mathbb{R}^3$ is the position of the Zero Moment Point (ZMP), it is shorthand for $\mathbf{t}_{\text{ZMP}}$.

$\mathbf{r} \in \mathbb{R}^3$ is the position of a foot, it is shorthand for $\mathbf{t}_{\text{foot}}$.

$\hat{\mathbf{c}} \in \mathbb{R}^6$ is the state vector of the linear model in the chapter on walking such that $\hat{\mathbf{c}} = [c^{\text{x}}\ \dot{c}^{\text{x}}\ \ddot{c}^{\text{x}}\ c^{y}\ \dot{c}^{\text{y}}\ \ddot{c}^{\text{y}}]^\top$.

$\hat{\mathbf{f}} \in \mathbb{R}^4$ is a particular vector of external forces and torques used for the linear model in the chapter on walking such that $\hat{\mathbf{f}} = [n^{\text{y}}_{\text{ext}}\ f^{\text{x}}_{\text{ext}}\ n^{\text{x}}_{\text{ext}}\ f^{\text{y}}_{\text{ext}}]^\top$.

$\mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{E}$ are matrices used for defining the linear model in the chapter on walking.

k as a subscript of the matrices and vectors in the linear model in the chapter on walking denotes a discrete time interval.

N as a subscript denotes the maximum length of the preview horizon of the MPC.

$\tilde{\mathbf{c}} \in \mathbb{R}^{6\text{N}}$ is the state vector of the MPC model in the chapter on walking such that $\tilde{\mathbf{c}} = [\hat{\mathbf{c}}_1^\top \dots \hat{\mathbf{c}}_{\text{N}}^\top]^\top$ over an N-step preview horizon.

$\tilde{\mathbf{u}} \in \mathbb{R}^{2\text{N}}$ is the control vector of the MPC model in the chapter on walking such that $\tilde{\mathbf{u}} = [\dddot{\mathbf{c}}_0^{\text{x,y}\top} \dots \dddot{\mathbf{c}}_{\text{N-1}}^{\text{x,y}\top}]^\top$ over an N-step preview horizon.

$\tilde{\mathbf{f}} \in \mathbb{R}^{4\text{N}}$ is the vector of external wrenches of the MPC model in the chapter on walking such that $\tilde{\mathbf{f}} = [\hat{\mathbf{f}}_1^\top \dots \hat{\mathbf{f}}_{\text{N}}^\top]^\top$ over an N-step preview horizon.

$\tilde{\mathbf{z}} \in \mathbb{R}^{2\text{N}}$ is the vector of ZMPs of the MPC model in the chapter on walking such that $\tilde{\mathbf{z}} = [\mathbf{z}_1^{\text{x,y}\top} \dots \mathbf{z}_{\text{N}}^{\text{x,y}\top}]^\top$ over an N-step preview horizon.

$\mathbf{U_c}, \mathbf{U_u}, \mathbf{O_c}, \mathbf{O_u}, \mathbf{O_f}$ are matrices resulting from condensing the Model Predictive Control Problem in the chapter on walking.

M as a subscript denotes the maximum number of variable footsteps in the preview horizon of the MPC.

$\check{\mathbf{r}} \in \mathbb{R}^{2\text{M}}$ is the vector of foot landing positions relative to the preceding foot position such that $\check{\mathbf{r}} = [{}^{\mathbf{r}_0}\mathbf{r}_1^{\text{x,y}\top} \dots {}^{\mathbf{r}_{\text{M-1}}}\mathbf{r}_{\text{M}}^{\text{x,y}\top}]^\top$.

$\check{\mathbf{z}} \in \mathbb{R}^{2\text{N}}$ is the vector of ZMPs relative to the supporting foot/feet such that $\check{\mathbf{z}} = [{}^{\mathbf{r}_1}\mathbf{z}_1^{\text{x,y}\top} \dots {}^{\mathbf{r}_{\text{N}}}\mathbf{z}_{\text{N}}^{\text{x,y}\top}]^\top$.

$\mathbf{S}$ are matrices used to concisely describe the MPC states and outputs.

$\mathbf{s}$ are vectors used to concisely describe the MPC states and outputs.

$\mathbf{v_{r_0}} \in \mathbb{R}^{2N}$ is a vector of fixed foot positions used to express the ZMP and variable foot positions in convenient local frames.

$\mathbf{V_r}, \mathbf{V_z}$ are matrices containing a certain structure of the foot rotations such that the ZMP and foot positions can be expressed in convenient local frames.

$\mathbf{g} \in \mathbb{R}^3$ is the acceleration vector due to gravity.

Δt denotes a discrete time step.

Introduction

We live in an amazing time for robotics, with continuous developments coming from various areas. Examples include: self-driving cars being continuously tested and legalized; small unmanned-aerial vehicles (UAVs) operating in several application areas, including the delivery of goods; *collaborative* industrial robots, working safely beside humans in manufacturing factories; and the recently concluded DARPA Robotics Challenge (DRC), showcasing several humanoid robots competing in a *real-world* scenario. This thesis is a contribution towards the convergence of the last two areas: humanoid robots with the ability of working together with humans.

Humanoid robots are robots with an anthropomorphic form. However, this term is often used loosely, since the robot can have varying degrees of similarity with a human. For this thesis, a *humanoid* is defined as a robot having two legs and two arms, that are attached to a torso with a head, all of which are functional. The functionality of the whole body is the key, since we will be utilizing it throughout this thesis. This definition also implies that humanoids are at the convergence of several robotics research areas: legged locomotion, manipulation control for the arms and hands, sensor processing, just to name a few. Although humanoids have long been conceptualized (especially in science fiction), it was only in recent decades that fully functional humanoids have started to become widespread, with a lot of variation between different classes. Some examples are shown in Fig. 1. The diversity is evident but the similarity in form allows us to be generic in our approach to programming humanoids such as these, which is one of the main goals of this thesis. Although these prototypes are certainly impressive, current day humanoids are still a long way from achieving the functionalities that we desire or envision. Since the form is human-like, the performance of tasks is similarly expected to be *human-like*. Unfortunately, this is not generally the case in the current state of the art. Typically, most of the robots in Figure 1 are only used within research laboratories, and therefore operate in structured and controlled conditions. A more realistic scenario is that of the DRC, where the winning team required less than 45 minutes for completing a required course, with many teams failing to even complete the first few tasks. Although the performance of the top teams was very impressive by current standards, a human can likely realize the same tasks in under 10 minutes. Clearly, much work is to be done before the general adoption of humanoids will take place.

While the technology for humanoids slowly matured and developed within re-

Figure 1: Some humanoid robots from recent years, from left to right: Boston Dynamics' Atlas, Kawada's HRP4, Honda's ASIMO, KAIST's HUBO, PAL Robotics' REEM-C, DLR's TORO, Aldebaran's Romeo.

search labs, industry has long adapted robots for manufacturing. Many of these specialized, fixed-base robots are big, powerful machines which are designed to be efficient and fast. However, as a consequence, they need to be isolated because of the inherent danger they represent. Oftentimes, these robots require their own workspace or even safety cages. This impedes them from collaborating alongside humans. However, in recent years, there has been a significant emerging branch of industrial robots that brought along a paradigm shift: *collaborative robots* that are designed to safely work side-by-side with humans. Although these are slow, in comparison to *classic* industrial robots, they are still efficient and reliable for certain tasks. Furthermore, they are designed to be easily reconfigurable (as opposed to being specialized) for a variety of tasks. Currently, the technology for this has become mature enough, and is starting to be commercialized by several companies, with Fig. 2 showing some examples. Although some companies adopt the single-arm design of *classical* industrial robots, others have chosen a dual-arm design, in some cases even including a functional head. The design choice of having two arms and a head seems to have stemmed from copying the upper-body humanoid design, or perhaps to help their applicability for working together with humans. Whatever the case may be for the design choice, these robots only need functional legs to be formally called humanoids. However, adding these legs has a lot of implications on how the robot must be controlled and programmed, returning us to the main topic of this thesis: how do we reconnect collaborative and humanoid robotics?

As mentioned, this thesis focuses on collaborative humanoid robots. Although simply described with two words, it is both deep and broad, with the potential to be much more than an industrial tool: it is potentially a general purpose tool in various settings. As it is a broad topic, we begin by giving some background on all the aspects related to collaborative humanoids in Chapter 1. From this, we slowly focus on the main points to be discussed further in the next chapters. Chapter 2 starts with detailing collaboration as it relates to robot control. This eventually leads to utilizing vision and haptics, to enable better collaboration. Chapter 3 then goes into the details on the implications for adding legs to collaborative robots. More

Figure 2: Some of the industry-focused, "collaborative" robots, from left to right: KUKA's LBR-iiwa, Universal Robots' UR3, ABB's YuMi, Kawada's NEXTAGE, Rethink Robotics' Baxter.

precisely, it details a walking pattern generator designed for physical collaboration. Lastly, Chapter 4 deals with the whole-body control problem. Specifically, the task of collaboratively carrying an object together with a human is broken down and implemented within an optimization framework.

Chapter 1

Background and state-of-the-art

This thesis contributes towards humanoids that can collaborate with humans, ideally with a performance similar to that of another human. This involves several different areas of research. To give some background, Sect. 1.1 starts from physical interaction in general, before going specifically into collaboration. This section also reviews the application of these ideas in robot control, which is our end goal. Next, section 1.2 focuses on the state-of-the-art in humanoid control with a particular view on enabling physical interaction. Finally, section 1.3 reviews the most relevant works on human-humanoid collaboration.

1.1 Physical human-robot interaction and collaboration

Physical human-robot interaction (pHRI) is one of the fastest growing research areas in robotics. It is a very broad field, dealing with vastly different aspects, ranging from the human psychology and behavioral science to robot control. This thesis is skewed towards robot control, although the human aspect always needs to be accounted for. Thus, we start with a quick review of some aspects from cognitive and behavioral science in subsection 1.1.1. With this, some important terminology is defined. Afterwards, subsection 1.1.2 gives a more detailed review of robot control for pHRI.

1.1.1 Terminology with a cognitive and behavioral science perspective

Starting with a cognitive and behavioral science perspective is important to gain some grounding. Since our ideal is to make a robot as effective as a human, a good initial perspective can be obtained by looking at how humans act. The aim is to gain insight into the underlying principles needed to program a robot to successfully collaborate with humans. However, to be clear, the interest is not to do things

exactly as a human would, but rather to take inspiration and try to apply the core concepts into a humanoid robot.

Before going into collaboration in particular, it is useful to start from the more general: interaction. More precisely, we are concerned with *physical interaction*, since the goal is defined as requiring physical labour. In this thesis, we define physical interaction as the participation of multiple agents (humans and/or robots), jointly, in a given task. Note that simply participating does not imply working together. To clarify this, we adapt the taxonomy of such behaviors from [1]. Although the meanings are kept, we will slightly adapt different terms to better clarify the work targeted in this thesis. The taxonomy of [1] starts by classifying the task. This can be done in two different ways:

1. divisible vs. interactive task,

2. competitive vs. cooperative task.

For the first task taxonomy: a *divisible* task is such that agents are able to divide the work without fear of conflicts, whereas an *interactive* task must be done together and at the same time. For example, cleaning a room is divisible but moving heavy furniture (requiring two or more agents) is interactive. Note that divisible tasks can be effectively treated as a collection of independent subtasks. This can be taken advantage of in the context of robot programming, where the subtasks require minimal interaction with the human. In fact, collaborative industrial robots generally target such kind of tasks. On the contrary, this thesis is focused on interactive tasks, that must be done together by humans and robots.

For the second task taxonomy: a *competitive* task is such that agents work against each other, whereas *cooperative* tasks require them to work with each other. For example, a tug-of-war game is competitive while pushing a heavy object together is cooperative. This thesis is clearly concerned with cooperative tasks.

Another taxonomy from [1] concerns the behavior of the agents independently from the task. Although the paper defined three general classes with some subclasses, we firstly simplify this to be:

- competition vs. collaboration.

Note that it is similar to the second task taxonomy. But rather than the nature of the task itself, the agents' behaviors are highlighted. As for the task taxonomy, we are concerned with making the robot behave in a collaborative manner. Apart from this, the agents may also be classified according to their roles. Studies have shown the importance of specialization of roles in collaborative tasks [2]. Perhaps, the most common role distribution is leader-follower. A *leader* is defined as the agent having a clear and independent intention for doing the task. Additionally, the leader seeks to guide the follower in doing the task. A *follower* is the agent, whose intention is dependent on the leader. A follower cannot act on his own to do the task and must be guided by the leader's intention. However, a follower can be *pro-active* if

s/he acts based on a prediction of the leader's future intention. Contrary to the leader-follower role distribution, an equal-collaboration approach is also possible [3].

Apart from terminology on interaction and collaboration, we can also take inspiration from how humans function. For example, we can take inspiration from studies regarding how humans move, such as [4], or regarding how they use their senses for collaborating. Humans usually use three distinct percepts for communication: sight, touch, and hearing. These factors all play a role in communication during the collaborative task [2]. Additionally, there has been evidence that humans tend to merge these percepts [5, 6] rather than to use them individually.

1.1.2 Robot control for interaction and collaboration

Robot control is a mature field, already being commercialized heavily in industry. However, the principles needed for interacting and collaborating with a human are not yet established and are the subject of a lot of the research in pHRI. One of the main issues is safety. Although there has been a lot of recent push towards this, with a view towards standardization (under the ISO/TC 184/SC 2 for robots and robotic devices [7] for example), we are still in the starting stage before general adaptation. Another of the main issues is the efficiency of the robot, for example its proactivity in following behaviors [8–10]. This issue in particular has tried to draw inspiration from behavioral and cognitive science.

Due to its nature, physical human-robot interaction has largely relied on the use of haptic data (force/torque) for control. By extension, it is grounded in a lot of the core concepts of force control in robotics [11]. In particular, impedance control [12] has been largely adapted with various enhancements. For instance, variable impedance control is used for human-robot collaboration in [13, 14], with parameters obtained from human-human experiments. Similarly, a variable damping controller is defined in [15], using the derivative of the interaction force. A method for improving impedance control, consists in utilizing an estimate of the human's intended motion [16]. Another possibility is to have a model. For instance, in [8], a minimum jerk model of human motion, inspired from [4], is used to have a good guess of the human intention. This is then used as a desired trajectory within an impedance control framework. Another example is given in [9], where machine learning methods are used to obtain a model of the task, which is then utilized within an adaptive impedance control framework. Conversely, mechanical impedance was shown to be a good model of a human, stabilizing an unstable system [17].

In spite of this important research in haptics, it is clear that vision can provide a lot of complementary information, that cannot be obtained from force/torque sensors alone. For example, information about object motion and human gestures/pose may be acquired. However, the integration of such information into human-robot haptic joint actions is not straightforward. The main issues are: what information would be helpful? how can this information be reliably obtained in the context of the task and platform? and how/where should this information be used? Since the priority

should still be interaction force regulation, the last issue is particularly important and implies the need for combining vision with force data. However, there have not been many applications of this in pHRI. To our knowledge, there has been no previous work on combining vision and force in the context of human-robot haptic joint actions. However, previous works on combining vision and force for object manipulation tasks can be found in the literature. A brief review of these methods follows.

The different approaches to combining vision and force information for manipulation tasks can be classified into three general categories [18]. These are: traded, hybrid and shared. The simplest strategy is termed as *traded* since the final control output is traded between a purely force-based controller and a purely vision-based controller. The switching between controllers depends on the task execution. A common example of this strategy starts by doing a *guarded move* visual servoing phase. A contact event causes the guarded move phase to stop. After this, the task is completed by force control.

The second category is that of *hybrid* methods. These are hybrid in the sense that the vision-based controller and force-based controller act at the same time, but in different spaces. This requires prior specification of a *task-frame* [19–21]. The task frame is a Cartesian reference frame that can be divided into vision-controlled and force-controlled directions. Doing this decouples vision and force into orthogonal spaces. Afterwards, the controllers can be designed separately and work independently in their own predefined space. The third and final category is termed as *shared* methods. These are so-called since there is no separation in time (the case of traded control) or in space (the case of hybrid control). The control responsibility is shared by both vision and force throughout the operation. By doing this, all available information can be used [18]. An example of a shared method is described in [22], where force feedback is used to correct the visual servo control trajectory.

In the context of human-robot haptic joint actions, a shared method is needed. With the human in the loop, safety is always a top priority. Therefore, the control must always be compliant in all degrees of freedom (DOFs). This can be achieved by always making use of the force information in all DOFs. Therefore, in order to make use of visual data, a shared control strategy must be devised. A good candidate for this is the impedance control framework [12]. Impedance control allows a manipulator to be compliant by defining a virtual mechanical impedance. Vision can be used in this framework to provide a reference trajectory that is tracked in the absence of external forces [23, 24]. When contact does occur, it has the properties of shared control methods, where vision and force determine the control of the same degree of freedom (DOF) simultaneously. This approach is preferred over the others since it can allow for compliance in all DOF. Previous works using this methodology [23–25] presented experiments of a robot interacting with objects. Here, we use this approach for physical human-humanoid collaboration. Having a human as a physical collaborator implies that some issues of this framework are to be revisited, typically the implication of impedance parameters, and the way vision

and haptics are combined in the context of physical collaboration.

Contrary to combining vision and force in control, another possibility is to fuse the different information sources. This can be done with sensor fusion techniques, for example cartesian frames are fused in [26], and fusion of proprioception and vision is done with an extended Kalman filter (EKF) in [27] or with task frame estimation in [28]. These approaches relate back to the merging of percepts by humans in [5, 6]. Although there is some merit to this, inherently there is a resolvability issue [29]. Force/torque and position/orientation(vision) are inherently different quantities. More formally, motion and force can be represented as dual vector spaces [30]. Another possible usage of vision is in trying to infer/guess the underlying intention behind motion. This concept was explored in [31] for a table-tennis playing robot (also an example of a competitive task). Another example is provided in [32], where the human hand is tracked, with vision, during a hand-over task.

1.2 Humanoid robot control

Humanoid robotics focuses on the design of robots directly inspired by human capabilities. This design gives many advantages when working together with humans in performing tasks. Because of this, humanoids are ideal research platforms for physical Human-Robot Interaction (pHRI). Typically, humans have extensive experience in physically collaborating with each other. Humanoid robots simplify such interactions, since they possess a human-like range of motion and sensing capabilities. These can be used to create suitable behaviors, by reducing the need for the human cooperator to learn how to interact with the robot. Although this goal is clear, many challenges are still present in the various research areas.

Among several issues that must be handled for humanoid robots, there are two that make them significantly different from those more commonly found in industry. The first concerns *whole-body control*. Often, a humanoid robot will have more DOF than an industrial manipulator. This allows several objectives to be realized at the same time. Furthermore, these objectives may have different priorities associated to them. To do this, optimization frameworks have shown to be effective [33–36]. We take this approach and adapt it here to our problem. This is detailed more in chapter 4.

The second differentiator is mobility in the environment. In contrast with fixed-base manipulators, humanoids have the capacity to freely move around. However, this advantage comes at a price since the control must now handle the balance of the robot. This is discussed in detail in the next two subsections, with subsection 1.2.1 giving a short review on the basic principles and terminology of humanoid walking. The main review is given in subsection 1.2.2, which focuses specifically on walking with physical interaction.

1.2.1 Locomotion and walking principles

Locomotion is the process of moving from one place to another. A robot that can 'locomote' becomes fundamentally different from the *fixed-base* robots commonly found in industry. The most obvious difference is the ability to move to different locations by itself. Formally, this ability is described with a *floating base* [30] that defines the pose of the robot base in a global reference frame. Effectively, it increases the configuration space of the robot. This gives it a significant advantage over fixed base robots. However, this advantage often comes with a drawback. Oftentimes, the floating base is not directly actuated. Therefore, to effectively control the floating base, we must use the contact forces with the environment. This is explained more formally at the beginning of chapter 3. However, if the floating base is not controlled properly, the robot will *fall*. This is what we want to avoid when locomoting in the environment.

Locomotion is too broad a term, it includes: walking, running, jumping, hopping, crawling, swimming, etc. Among these, walking is the most common and most useful for doing physical collaboration. It has been one of the main subjects of research for humanoid robotics and legged robots in general. A lot of research has been placed into it and there are good reference texts that describe the core principles of robot walking [37, 38]. Before moving into a detailed review on walking related to physical interaction, we shall review some commonly used concepts and terms from the literature.

One of the common methodologies in making robots walk is to start from a *reduced model*, instead of directly handling the complexity of the whole-body dynamics. A detailed explanation and analysis of the dynamics of the reduced model can be found in [39]. The reduced model is often based on the Center of Mass (CoM). Since contacts are essential in locomotion, the Center of Pressure (CoP) is another important point. However, in the robotics literature, the term Zero-Moment Point (ZMP) is more prevalent. In the normal case, the CoP and ZMP are equivalent [40]. These two points encapsulate the essential dynamics of the system, so that the reduced model is represented by an equation relating these two. Another important concept related to the ZMP is the support polygon. This is defined as the convex hull of the contact points of the feet on the ground. It defines the area where the ZMP should stay in order for the robot to remain balanced. One of the common benchmark reduced models is the linearized inverted pendulum (LIP) model [41], where the relation is given a physical meaning. A lot of variants to this are the subject of other research. This includes adding a spring to the model to account for energy storage, adding a flywheel to account for angular momentum effects [42], or attempting to generalize the ZMP to non-coplanar contacts [43, 44]. Part of our work is concerned with a reduced model that accounts for physical interaction. Works related to this are mentioned later on.

Apart form the CoM and ZMP, another more recent concept is the capture point [45, 46], which stemmed from capturability analysis. Since having a general

definition of a fall is difficult, capturability seeks to reduce it to being able to come to a stop in a certain number of steps. The capture point can also be shown to be related to the ZMP [38].

With the reduced model, we need to control the robot. This area is also well researched and a widely-recognized benchmark is preview control of the ZMP, with pre-defined future footstep locations [41]. One of the latest improvements on this method is a model predictive control (MPC) approach, with constraints on the ZMP, and variable footsteps that are part of the optimization result [47]. Our current work is based on [47], with the foremost contribution that it accounts for, and uses, physical interaction (i.e. external sustained forces).

1.2.2 Walking with physical interaction

Because of the availability of numerous walking algorithms, some works have tried to take these algorithms as they are and adapt them to more complex scenarios. These works are reviewed here.

In physical collaboration, we can take the example of transporting large or heavy objects together with a human [10, 48–50]. These were done using the methods described in [41, 47]. In these works, the walking pattern generator (WPG) does not take into account physical interaction, which is treated as a disturbance. Since success relies on the robustness of the WPG to this disturbance, an impedance controller is added in the arms, to regulate the interaction forces.

To our knowledge, there have been no works where walking is specifically designed for use in human-humanoid collaboration. The closest to this in the literature takes into account external forces in the WPG model but stops there, effectively treating it as a modeled disturbance instead of a useful signal. Several works do this in demonstrations, where a humanoid robot pushes/pulls objects while walking [43,44,51–57]. All these works present different variations to improve [41], by considering the physical interaction with the object. The work in [52] has an interesting simplification: here, the robot pushes only during the double support phase (when both feet are in contact with the ground), to allow a larger support polygon (and equivalently stability margin) during pushing. To compensate the pushing forces on the hands, the desired ZMP is changed iteratively. However, this causes a discontinuity in the CoM trajectory, so the new CoM trajectory needs to be smoothly connected to the existing one [52]. This also led to research on a *generalized ZMP* [43, 44]. The work in [51] presents a preview controller of the ZMP, emphasizing *short cycle pattern generation*. One of the experiments shows a humanoid pushing a table. This was done by modifying the desired ZMP to take into account external forces in the ZMP computation (compensate a measured disturbance), and moving the torso by the ZMP error (compensating unknown/unmeasurable disturbances). The work in [53, 54] focuses on interacting with known objects, while compensating for unknown effects such as friction. This was demonstrated by pushing a wheelchair and opening cabinets, doors, etc. Similar to other works, the ZMP computation takes into account

the reaction forces in the hands. Additionally, a footstep modifier was introduced using heuristics from the difference between hand and feet operational points and a ZMP error. Meanwhile, in [55, 56], the main goal is to consider a manipulability criteria for the hands and a balance criteria for the legs. Experiments show the robot pushing a table while walking with varying "resistance" (making it heavier, adding brakes, or with a human at the other end). Again, the ZMP is modified, but this time it also takes into consideration the CoM projection in the support polygon. Furthermore, a theoretical limit on the maximum hand forces is obtained by considering the constraint of ZMP in the support polygon (to retain dynamic balance). Yet another application is in the teleoperation of a humanoid robot [57]. The interface abstracts the WPG, and again the ZMP takes into account the reaction forces in the hands, with the addition of a smoothness term. Furthermore, the unknown force is modeled by a linear damping effect. All these works rely on simplifying assumptions on the interaction wrench applied on the robot, whereas the full wrench is presented and utilized in chapter 3. Additionally, all of these works, except for [51], mention the use of impedance/force control on the arms to increase robustness. In [58], the effect of this added impedance control is directly taken into account in model predictive control (MPC), although it is used for postural stability rather than for walking. Another closely related work is [59], which builds on [53, 54] in pushing heavy objects. In particular, the range of allowable external forces is obtained from the model, considering a pre-defined ZMP, as in [41]. Differently from this, the MPC framework presented in this thesis allows to vary both the external forces and ZMP (depending on the objective function weights). Another set of closely related works proposes 'stepping to maintain balance' [60–63]. However, these works consider the external wrench as transient, rather than sustained throughout the task.

To summarize, there is not much existing literature on the topic of walking under sustained external forces. A lot of research in walking is validated for abrupt external disturbances, but not for sustained ones. However, in the existing works listed above, these are the common points:

- Although some are presented as *whole-body* frameworks, the walking task is always abstracted from the hand control tasks, and some type of compensation is done to maintain dynamic balance. That is, the force on the hands is treated as a disturbance to be rejected.

- Most works used a form of compliant control in the contact point (usually the hands) to make sure the net forces are low.

1.3 Human-humanoid collaboration

Probably the best example for interactive, cooperative tasks is the transportation of large and/or heavy objects together. Early work on this topic was done in the Humanoid Robotics Project (HRP), where the HRP-2P humanoid cooperates with

a human for a panel transportation task [48]. This was envisioned to have possible future applications in construction sites. The same scenario can also be applied to the household, such as moving furniture (e.g. table).

Earlier work in this topic involved *mobile manipulators* [64]. *Mobile manipulators* use wheels instead of legs for locomotion. The trade-off is having an easier control (from wheeled robotics technology) over more limited mobility (i.e. requiring drivable terrain). Although the locomotion mode is different, this revealed one of the main issues early on: how to coordinate the motion of the mobile/floating base with the manipulator motion of the upper body. A more recent example of a mobile manipulator doing collaborative carrying is presented in [65].

A similar task has also been tested on smaller scale humanoid robots. For example in [66], the NAO humanoid was used in collaborative tasks. The main interest is in using other internal sensors in place of force/torque sensors in the hands, which are generally used in the works with full-scale humanoids. NAO is also used in [67], where the capture point is used to guide walking. Another example is Darwin robots collaboratively carrying a stretcher [68]. However here, the human element is removed. And with only robots, the interest is turned to synchronizing the robotic agents.

Within our research group, human-humanoid collaboration has been one of the main topics starting with [69]. One of the foremost contribution of this work is homotopy switching between leader and follower controllers [70]. After this, [71] explored the idea of studying human-human dyads to try and understand how they cooperate for such a task [10]. The main interest was to enhance impedance control with some proactivity. Eventually, some preliminary user studies were also done [71]. This was interesting to see reactions from the *common user* (i.e. someone not involved with robotics research) and just how applicable such a technology would be in the future. In [49], the idea of role switching and proactivity while turning, is also explored.

From this review on the state-of-the-art in human-humanoid collaboration, we can see that there is a lot of work needed to be done in several different aspects. In particular, this thesis concentrates on 3 main aspects which follow in the corresponding chapters. Chapter 2 explores the integration of vision with haptics in a collaboration task. Chapter 3 goes into how walking pattern generators can be designed specifically for the expected physical interaction. And finally Chapter 4 shows how we break down the collaborative carrying task starting from a taxonomy of human dyads towards whole-body control with a humanoid robot.

Chapter 2

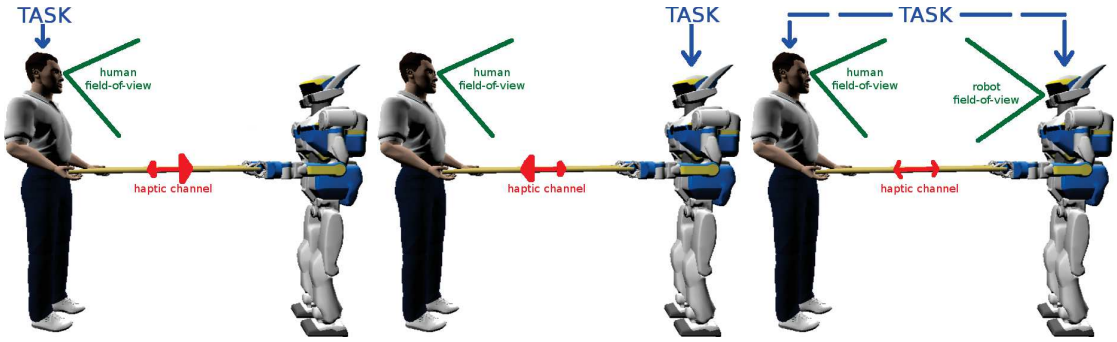
Using vision and haptic sensing in physical collaboration

Haptic joint actions are interactive, cooperative tasks, requiring a haptic interaction throughout the task. This applies to several collaborative tasks that humans do together. For example: helping someone to their feet, handing over objects, supporting someone while walking, etc. Another common example is carrying a large object together. As haptics is a requirement for all these tasks, most works rely on it to be able to control the robot.

Perhaps the most common example of haptic joint actions is carrying an object together with a robot, such that it has become a benchmark for studying the underlying principles of physical human-robot collaboration. One of the main principles targeted by research is the allocation of roles between the agents, and specifically how a leader-follower behavior is achieved. This has often been studied with haptics, because of the nature of the task. In this chapter, we reconsider the collaborative carrying task when vision can be used to improve it. Section 2.1 details the framework we designed, to add vision to haptics in collaborative tasks. This framework is applied in the case studies of section 2.2. Section 2.3 concludes the chapter with some results and discussion.

2.1 Framework for incorporating vision into haptic joint actions

Within our research group, there has been a previous focus on physical collaboration using only force/torque sensors [69,71]. These works have led to a good understanding of the state-of-the art on programming humanoid robots to physically collaborate with humans. The focus is a particular example: collaboratively carrying large objects, which has been explored for its practicability and potential applications (e.g. in construction [48]). To improve on these, we first tried to enhance the previous results by adding vision data. Fig. 2.1 illustrates the three important information



(a) robot as pure follower (b) robot as pure leader (c) equal collaboration

Figure 2.1: Cases of human-robot haptic joint actions

sources to be used: haptic information, visual information and prior task knowledge. In this example, the haptic interaction exists through the object - the haptic channel. This means that a force/torque applied on one end of the object is felt by the partner on the other end. Because of this, previous research has focused primarily on regulating interaction forces for safety. The use of vision for the robot has not been investigated before in this context, and is the main focus here. Finally, the prior task knowledge can be used as a guideline on how vision and/or haptic information should be used for the task.

Although what can be achieved using force data alone (i.e., by a *blind* robot) is impressive, vision is clearly required for some tasks, and could possibly help make the robot proactive. For example, in collaboratively carrying a table, force data alone cannot be used to see if an object on top of the table is about to fall down. It is clear that visual information is largely complementary to force information (analogous to the human senses of sight and touch). Combining these might enable a humanoid to perform more complicated tasks, similar to a human combining these percepts [5].

In the context of role allocation, the most commonly studied scenario of human-robot collaboration consists in making the robot a pure follower as in Fig. 2.1(a). This figure illustrates that the task is only known to the human leader a priori. Through the haptic channel, the human communicates his intentions to the robot. With a properly designed controller that takes into account interaction forces, the robot can follow the human's lead [10]. This is represented by the bigger arrow in the haptic channel from human→robot. Furthermore, the human is able to obtain visual and haptic data as to how good/bad the robot follows the task.

Another situation is the exact opposite of the robot being a pure follower, i.e., it can be the leader of the task. This is illustrated in Fig. 2.1(b). An example of such a scenario is presented in [49], where a joystick is used to give the robot direct information on the task. Although a second human provides this task information via the joystick, in the context of collaborative carrying, the robot is the leader. Without knowing the task of the robot, the human partner then tries to help the

robot carry the object to a desired location. It is fair to assume that the human uses both the sense of sight and touch to achieve this task. Apart from the two illustrated examples, other possibilities exist. For example, a combined leader-follower strategy has also been developed [3, 70]. The concept for this strategy is that a sharing of roles can be possible where one is both leader and follower (with varying degrees).

Finally, it can be noticed that, in these previous works, the robot is solely reliant on haptic information. No visual data is used in the task, as illustrated in Fig. 2.1(a,b) by the limitations of the robot field-of-view. The aim of the work here is to move towards the general case of Fig. 2.1(c), particularly adding vision as another information channel. In order to do this, the main question is how does one combine vision and force information in the context of haptic joint actions. The context presents significant challenges, specifically by having a human in the loop.

Our general approach to combining vision and haptic cues consists in coupling a visual servoing controller to an impedance controller. This simplifies the design, by decoupling the vision and force controllers in a systematic way. An overview of the complete control framework is shown in Fig. 2.2. Furthermore, it also shows the task example used here - balancing an object on the table. The following subsections explain this general framework in a bottom-up approach starting from the lower level controllers and abstracting it higher to the cognitive level. The lowest level of control is the inner joint-level control. This is represented by $\mathbf{q}$ in Fig. 2.2. To abstract from the joint level to the *task level*, the Stack-of-Tasks framework is used [33]. It is a generalized inverse kinematics abstraction layer that creates a hierarchical organization of different tasks to be executed giving higher priority to critical tasks [33]. It allows for easier integration with sub-tasks. For example, our experiments make use of the walking algorithm in [47] as a sub-task. Later on in this thesis, we go deeper into these two points: namely *walking* in chapter 3, and *whole body control* in chapter 4. For now, the subtasks relevant to this chapter are explained in the corresponding subsections.

2.1.1 Impedance control

The first sub-task concerns the hands/grippers. In Fig. 2.2 the humanoid uses its grippers to co-manipulate an object with a human. To do this, it needs to be safe and intuitive to use. Here, impedance control [12] is used to regulate the contact interaction (for safety) between the robot and its environment. Impedance control is based on a simple physical analogy to a virtual mass-spring-damper system [12]. This system is governed by the general equation:

$$\mathbf{h} = \mathbf{G}_{\mathrm{m}}(\ddot{\mathbf{p}}_{\mathrm{des}} - \ddot{\mathbf{p}}) + \mathbf{G}_{\mathrm{b}}(\dot{\mathbf{p}}_{\mathrm{des}} - \dot{\mathbf{p}}) + \mathbf{G}_{\mathrm{k}}(\mathbf{p}_{\mathrm{des}} - \mathbf{p}). \tag{2.1}$$

The contact interaction is measured by the force-torque sensors in the robot grippers and is represented as $\mathbf{h}$. The vectors $\mathbf{p}_{\mathrm{des}}, \dot{\mathbf{p}}_{\mathrm{des}}$ and $\ddot{\mathbf{p}}_{\mathrm{des}}$ indicate a desired pose and its first and second derivatives. Correspondingly, vectors $\mathbf{p}, \dot{\mathbf{p}}$ and $\ddot{\mathbf{p}}$ represent the corresponding actual pose, and its first and second derivatives. Finally, matrices

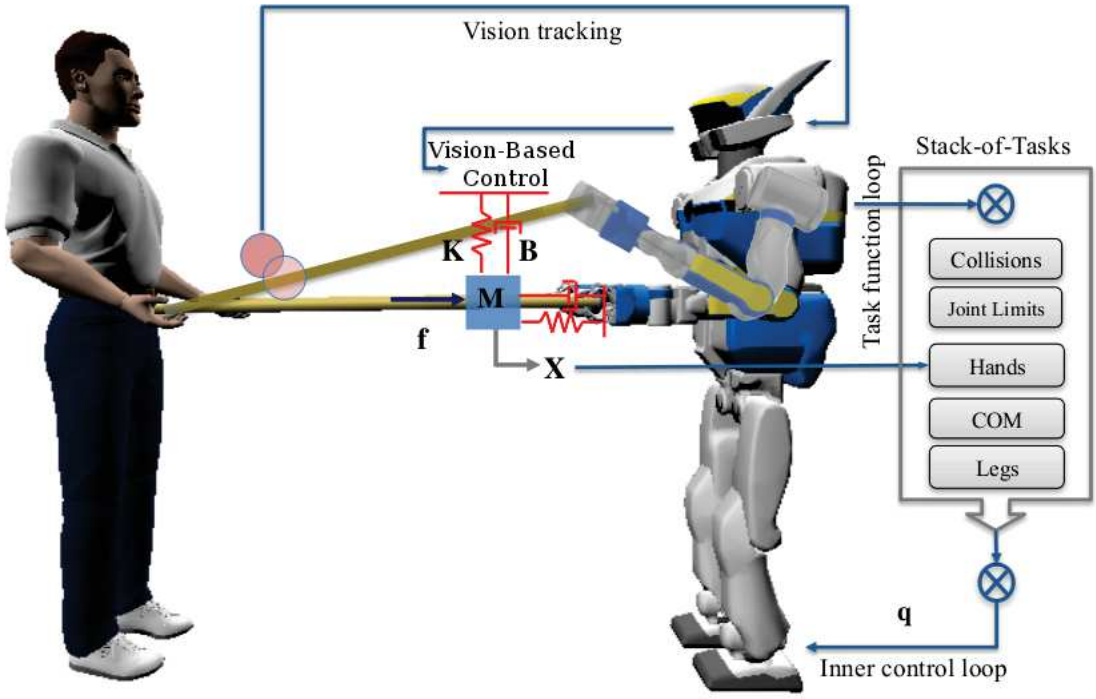


Figure 2.2: An illustration of the overall workflow of the algorithms running on the robot to enable collaboration with the human, using both vision and force data.

$\mathbf{G}_\mathrm{m}, \mathbf{G}_\mathrm{b}$ and $\mathbf{G}_\mathrm{k}$ are the inertia, damping and stiffness parameters that define the desired virtual mass-spring-damper system [12]. Strictly following the terminology and causality from [11, 12], our implementation is an *admittance controller*, since the robot is position-controlled by the Stack-of-Tasks, which uses the output of $\mathbf{p}, \dot{\mathbf{p}}$ and $\ddot{\mathbf{p}}$ from the impedance controller. These are obtained by solving the differential equation of Eq. (2.1), given the other variables. The parameters $\mathbf{G}_\mathrm{m}, \mathbf{G}_\mathrm{b}$ and $\mathbf{G}_\mathrm{k}$ are determined empirically to provide comfort for the human collaborator. The desired pose and trajectory of the mass-spring-damper's reference position, $\mathbf{p}_\mathrm{des}, \mathbf{p}_\mathrm{des}$, and $\ddot{\mathbf{p}}_\mathrm{des}$, are detailed in the next subsection.

2.1.2 Proactive behavior and visual servoing

For the general impedance controller of Eq. (2.1), a *passive* behavior is defined by setting the desired pose $\mathbf{p}_\mathrm{des}$ as constant. This case is illustrated in Fig. 2.1(a) where only the human knows about the task to be done. This is the *classic* case in human-robot collaboration. In such a case (and considering constant impedance parameters $\mathbf{G}_\mathrm{m}, \mathbf{G}_\mathrm{b}, \mathbf{G}_\mathrm{k}$), the robot motion $(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}})$ can only be initiated by an external wrench $\mathbf{h}$ due to Eq. (2.1) . Recent research aims at making the robot a proactive follower, to make the system more comfortable for the human. A way to achieve this is by creating a suitable desired pose and trajectory $(\mathbf{p}_\mathrm{des}, \dot{\mathbf{p}}_\mathrm{des}, \ddot{\mathbf{p}}_\mathrm{des})$, such that the human

effort is minimized [8, 10]. These works differ in the approach taken to produce the desired pose and trajectory. In [8], human motion is predicted by a minimum jerk model to give the desired pose. In [10], a human pair doing a joint transportation task was studied, and it was observed from the data that the pair moves in constant velocity phases during this task. A finite state machine (FSM) is then designed by using the constant velocity assumption, giving the desired pose and trajectory. Haptic cues are used to determine the switching of states in the FSM [10].

We take a different approach here, as illustrated by Fig. 2.1(c). Here, the humanoid is given knowledge of the task. This is done by designing a visual servoing controller specific to the task and using the output as the desired trajectory $(\mathbf{p}_{\text{des}}, \dot{\mathbf{p}}_{\text{des}}, \ddot{\mathbf{p}}_{\text{des}})$ of the impedance controller. This also means that the robot has some autonomy in doing the task, driven by its own knowledge of the state of the task. With the reasonable assumption that, during the collaborative task, human motion is task-driven, the source (human intention to do the task) is taken into account rather than the result (human motion). This differentiates our approach from those that aim to model/predict human motion such as in [8, 10].

Visual servoing consists in controlling robots using visual information [72, 73]. To create the visual servoing portion of the framework, two important components are needed: visual feature tracking and a controller based on this feature [72]. However, in the current state-of-the-art, for both modules there is no *best* approach that fits all tasks and problems. Existing methods have important trade-offs to consider for the whole system [72]. In our works, we take an analytic approach to building the visual servoing portion. These will be detailed for each of the problems in the case studies of section 2.2.

2.1.3 Visual tracking

To start the whole control framework, visual information needs to be processed. In the example depicted in Fig. 2.2, this algorithm gives data about the ball on top of the table. For our case studies, the raw data is a combination of an RGB image and of a depth map, obtained from an RGB-D sensor mounted as the *eyes* of the humanoid. This raw data needs to be processed into an estimate of the pose or position to be controlled by the vision-based controller. This is done by tracking a salient visual feature throughout the image sequence, and extracting the needed pose or position information from this. Although there is not yet any generic visual tracking algorithm that can work for any and all cases, well-known computer vision methods from reference texts, such as [74–76], may be used to obtain this information, given some simplifications. Furthermore, some knowledge of the task is needed to know what is the important visual information that is needed. This is the reason why the case studies presented here use trivial objects - a cube of known size and color, and a ball of known color. Although far from generic, visual tracking is a fairly well-developed field in computer vision, and a wide range of algorithms have been developed, as reported in this extensive survey [77]. Similar to visual servoing,

the vision algorithm, applied in each of our case studies, is detailed in section 2.2.

2.2 Case studies

To test the framework described, we tackle two case studies of human-robot haptic joint actions, that clearly benefit from visual information. Because the task of *collaborative table-carrying* has been well-studied within our research group, this is used as the base task. An object is then placed on top of the table and the additional task is concerned with this object, and with the table tilt angles $(\phi_{\mathrm{x}}, \phi_{\mathrm{y}})$. A simplified side-view of the task in Fig. 2.3 shows ϕ_{y} and its relation to the height difference z_r, and to the table length l_t. Furthermore, three important reference frames are drawn in this image, to facilitate the explanations that follow. These are: the control frame $\{cf\}$, a local reference frame on the robot $\{l\}$ and the table frame $\{t\}$. The control for the robot can be done just by defining the pose ${}^{l}\mathbf{T}_{cf}$. This is justified by assuming a rigid grasp during the whole task. This means that the pose of the right and left hands: ${}^{cf}\mathbf{T}_{rh}$ and ${}^{cf}\mathbf{T}_{lh}$ are constant throughout the task, and generating the 2-handed control merely consists in a change of frame. To achieve this, the hand poses $\{rh\}$ and $\{lh\}$ are controlled in the local frame according to:

$$ {}^{l}\mathbf{T}_{hand} = {}^{l}\mathbf{T}_{cf}\,{}^{cf}\mathbf{T}_{hand} \qquad hand = \{rh, lh\}. $$

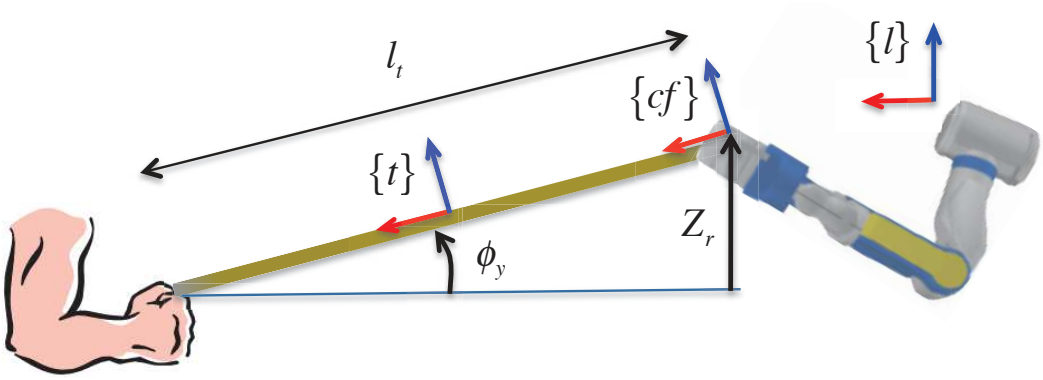


Figure 2.3: A simplified "thin beam" model, used to control the table tilt, through the height offset between robot and human grasps.

2.2.1 Stationary Object - keeping the plane level

In this scenario, a green cube is placed on top of the table as a representative *stationary object*. This scenario, along with the important reference frames, is shown in Fig. 2.4.

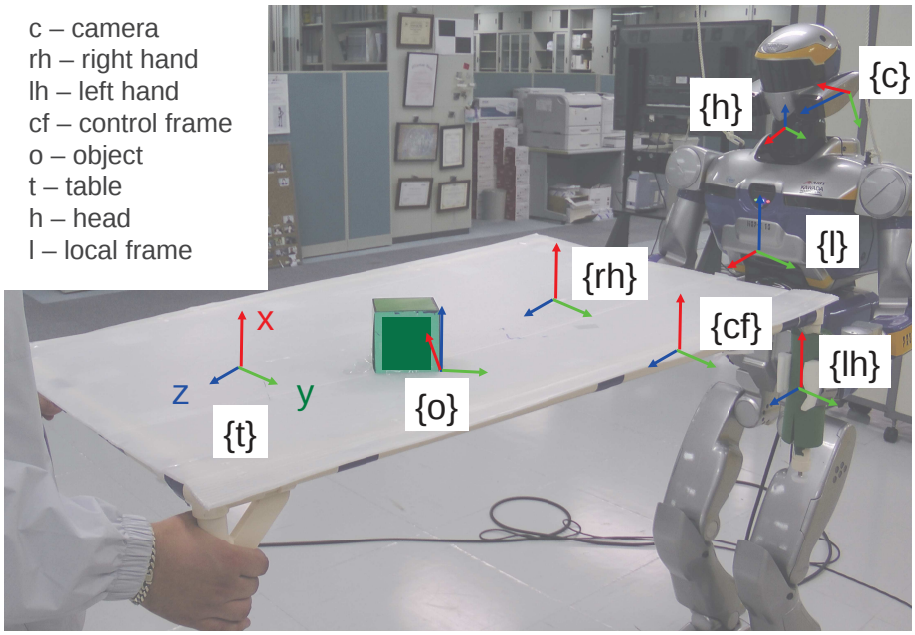


Figure 2.4: Case 1: table carrying scenario with stationary object.

With the simplified model of Fig. 2.3, vision is used to estimate ϕ_y – the inclination of the table. This is used as an error signal for a table height controller, to be detailed later. For this particular case, we only use the RGB data (not the depth) from the camera mounted on the humanoid robot head. Because of the viewpoint, ϕ_y cannot be directly observed on images taken from the robot, and must be extracted from 3D data. Although a variety of ways to extract this 3D data exist, a fast visual tracker is preferred here.

Visual tracking is used to obtain the pose of an object, of a priori known model, resting on the table (e.g., the cube of Fig. 2.4, with frame $\{o\}$ linked to it). The pose is represented by the homogeneous transformation matrix ${}^c\mathbf{T}_o$, where the camera frame $\{c\}$ is the reference. Frame $\{o\}$ is defined, so that its z-axis corresponds to the vector normal to the table/beam (see Fig. 2.4 and 2.3). This vector forms angle ϕ_y with the z-axis of frame $\{l\}$. To obtain the transform, a virtual visual servoing approach is used here for tracking and pose estimation [78]. This method relies on a model-based edge tracker, and is available as part of the visual servoing software library – ViSP [79]. It works by first initializing the projection of the object model onto the image. The edges are then tracked throughout the image, and a robust optimization process is used to obtain ${}^c\mathbf{T}_o$ from fitting the tracked edges onto the model [78]. A visualization of a typical result (${}^c\mathbf{T}_o$) is shown in Fig. 2.5 (left and middle images). Figure 2.5 also shows, in the rightmost image, how edges are tracked in the normal direction [78].

Reliability can be an issue for visual tracking and even state-of-the-art algorithms can fail [77]. This uncertainty is a problem, especially if the visual tracker output

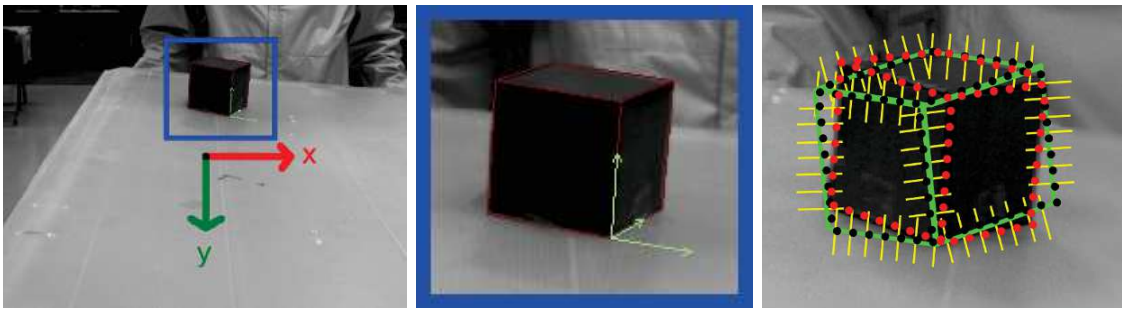


Figure 2.5: Typical result of the visual tracker. The full image is at the left. The middle image is a zoomed-in portion bordered by blue, with the projection of the cube's model in red, and the object frame in green. The right image shows how edges are tracked.

is to be used for control. Therefore, precautions are taken here to prevent this. A known platform-specific problem in humanoid robots is the motion induced by walking. The effect on the image is a characteristic oscillatory motion [80, 81]. Furthermore, impact forces resulting from the walk can cause significant blur on some images. These problems make it necessary to add robustness to the tracker.

Although it is possible to model the cause of these problems (walking) and compensate for it directly [80,81], a more general approach is taken here to handle other unforeseen disturbances. More specifically, a fault detection and tracker reinitialization process is implemented. This method is also advocated in [77] as a possible future trend, since all trackers can fail given a difficult enough condition.

The first requirement is the ability to detect a fault. The covariance matrix of the tracker optimization process is used as a *measure of goodness*. A *fault* condition arises if $\mathbf{var} > \mathbf{thr}$ where **var** is the variance vector (diagonal elements of the covariance matrix) and **thr** is a threshold vector that is manually tuned off-line by observing the tracker results from a typical image dataset. When one or more of the vector values is at fault, the tracker is reinitialized.

The next requirement is a reinitialization procedure using a good guess of ${}^{c}\mathbf{T}_{o}$. The reinitialization itself is trivial, and a method is already provided in ViSP. The main problem is estimating ${}^{c}\mathbf{T}_{o}$. A tracker failure often indicates that the assumption of continuity between images is invalid. Therefore, the data for reinitialization must come mainly from the current image. Another important consideration is the runtime. Obtaining a guess for ${}^{c}\mathbf{T}_{o}$ should be fast enough, so that continuity towards the next images can be safely assumed. Therefore, speed is chosen over generality.

To start, the object needs to be detected in the current image. This is done quickly, by thresholding and applying a sliding window for optimal detection. For thresholding, the hue space is used because the object used in this work has a distinct color. To speed up sliding window detection, the concept of image-pyramids is used, with *coarse detection* in a smaller scale version of the image. The result is used for

successively finer detections up to the original image size. This results in a good localization in image space ${}^{I}(x, y)$ where x and y are normalized image locations such that:

$$x = \frac{{}^{c}X_o}{{}^{c}Z_o} \qquad y = \frac{{}^{c}Y_o}{{}^{c}Z_o},$$

with ${}^{c}(X, Y, Z)_o$ the Cartesian coordinates of the object $\{o\}$ in the camera frame $\{c\}$ (see Fig. 2.4). A correct pose at the previous iteration $t - \Delta t$ (Δt is the control time step) can be used as a guess for the object orientation ${}^{c}\mathbf{R}_o^{t-\Delta t}$ and depth ${}^{c}Z_o^{t-\Delta t}$, so that the new pose, at the current iteration t, is defined as:

$${}^{c}\mathbf{t}_o^{t} = \begin{bmatrix} x^t \cdot {}^{c}Z_o^{t-\Delta t} \\ y^t \cdot {}^{c}Z_o^{t-\Delta t} \\ {}^{c}Z_o^{t-\Delta t} \end{bmatrix} \qquad {}^{c}\mathbf{R}_o^{t} = {}^{c}\mathbf{R}_o^{t-\Delta t}.$$

Although this new pose is imperfect, it is a good guess for reinitializing the tracker. Furthermore, the assumptions used here fit with the table carrying task done by a humanoid robot, namely: the depth to the object ${}^{c}Z_o$ is fairly constant, the rotation of the object is minimal, and most of the perturbation from walking results in a perturbation in image space ${}^{I}(x, y)$.

Lastly, another covariance check is done after the tracker is reinitialized. In the event that even the reinitialization fails, a failure signal is produced such that the visual servo controller also stops, thus preventing erroneous motions. This is more of a safety measure, since the tracker reinitialization worked well throughout the experiments.

Referring back to Fig. 2.3, ϕ_y is defined using $\{l\}$ as the reference. However, visual data gives ${}^{c}\mathbf{T}_o$, and as such a change of frame is needed:

$${}^{l}\mathbf{T}_o = {}^{l}\mathbf{T}_h \, {}^{h}\mathbf{T}_c \, {}^{c}\mathbf{T}_o. \tag{2.2}$$

${}^{h}\mathbf{T}_c$ is the pose of the camera in the robot's head frame. It is a constant matrix obtained from an off-line camera-robot calibration procedure. The pose of $\{h\}$ in the local frame (${}^{l}\mathbf{T}_h$) is available from proprioception. The angle ϕ_y can then be extracted from the rotation matrix of ${}^{l}\mathbf{T}_o$, i.e., ${}^{l}\mathbf{R}_o$, by

$$\phi_y = \arctan(-R_{13}, R_{33}), \tag{2.3}$$

where R_{ab} is the element at row a column b of ${}^{l}\mathbf{R}_o$. Eq.(2.3) is obtained from the relationship between axes that a rotation matrix represents. The z-axis of $\{o\}$ is the column vector where $b = 3$, since $\{l\}$ is the reference, the important components are in the x-axis ($a = 1$) and z-axis ($a = 3$). The final result ϕ_y, is only dependent on the rotations of Eq. (2.2) and the program implementation can be optimized as such. Furthermore, only results where $\frac{-\pi}{2} < \phi_y < \frac{\pi}{2}$ are considered valid. The limits correspond to the table being fully vertical and provide a safety measure.

Visual servoing enables the direct use of visual information in the controllers. To start the design, the error e needs to be defined. Fig. 2.3 shows that ϕ_y, the angle

between the table normal and the vertical is suitable such that: $e = \phi_y - \phi_y^*$, where ϕ_y^* denotes the desired value of ϕ_y. Defining a task that keeps the table horizontal implies that the desired value $\phi_y^* = 0$ making $e = \phi_y$. The model of the task can be defined as:

$$l_t \sin \phi_y = Z_r \tag{2.4}$$

Eq. (2.4) relates the observed angle ϕ_y to the height difference (Z_r) and the table length (l_t). Differentiating with respect to time and rearranging the terms results in:

$$\dot{\phi_y} = \frac{\dot{Z}_r}{l_t \cos \phi_y} \tag{2.5}$$

Eq. (2.5) is the model of the system and the controller can be derived from this. If, for example, an exponential decrease of the error is desired, it must be $\dot{e} = \dot{\phi_y} = -\lambda \phi_y$. Since the table length l_t is constant, it can be considered as part of the gain parameter λ. The control law then becomes:

$$\dot{Z}_r = -\lambda \hat{\phi_y} \cos \hat{\phi_y}, \tag{2.6}$$

where $\hat{\phi_y}$ represents the estimate of ϕ_y. If the estimation is perfect ($\hat{\phi_y} = \phi_y$), plugging (2.6) into (2.5) yields: $\dot{\phi_y} = -\frac{\lambda}{l_t}\phi_y$. This shows that l_t contributes only to the convergence speed, and as mentioned it is not necessary to know its value. It only affects the tuning of the gain λ.

To use this result in the impedance control framework, $\dot{Z}_r$ is numerically integrated such that Z_r at the current digital time step is obtained as: $Z_r^t = Z_r^{t-\Delta t} + \dot{Z}_r^t \Delta t$. Lastly, a constant velocity is assumed throughout the time step such that $\ddot{Z}_r = 0$. The results here ($Z_r, \dot{Z}_r, \ddot{Z}_r$) are then used as the Z part of $\mathbf{p}_{\text{des}}, \dot{\mathbf{p}}_{\text{des}}, \ddot{\mathbf{p}}_{\text{des}}$ in the impedance controller, taking into account the difference in reference frame (i.e. $\{l\}$ and $\{cf\}$). The results of this are shown in section 2.3.

2.2.2 Moving Object - keeping a ball from falling off

In this scenario, a ball is placed on top of the table as seen in Fig. 2.6. Any disturbance will tend to make the ball fall off the table. As in the previous case, two main components are needed: visual tracking and the control design which are detailed as follows.

This time, both RGB and depth data from the camera are used. The aim of the vision algorithm is to process this raw data into visual features that can be used for control. An error signal can be defined by ${}^t x_o - {}^t x_d$ and ${}^t y_o - {}^t y_d$. For the example task here, z is irrelevant, since ${}^t z_d \equiv {}^t z_o$. Since the desired location ${}^t(x, y)_d$ is arbitrarily defined, the vision algorithm only needs to obtain ${}^t(x, y)_o$. A variety of vision algorithms that can do this may be used, with speed as another consideration. For example, given the object model and the table model, it is possible to use a model based tracker, similar to the other case study. Designing a novel vision algorithm is

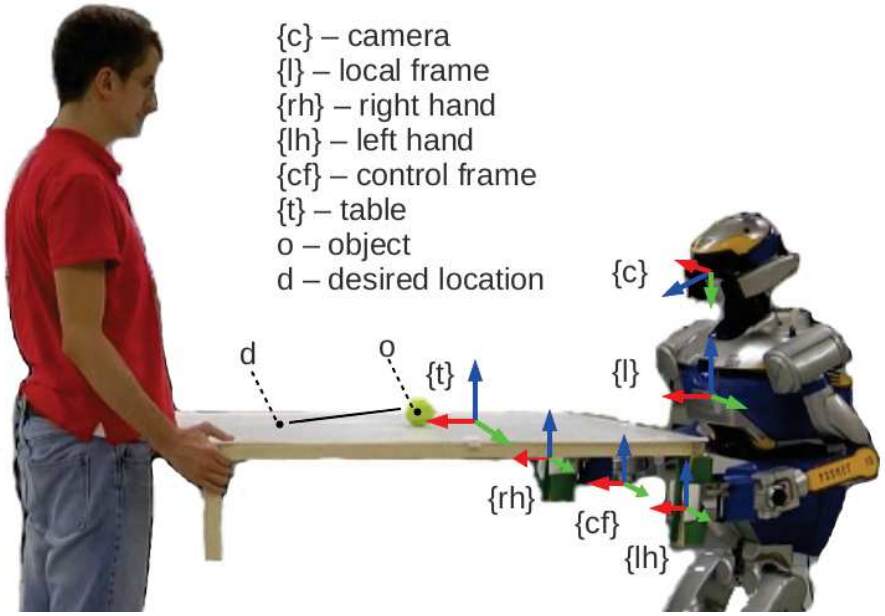


Figure 2.6: Case 2: table carrying with moving object

not the focus of this work, so we use well-known methods [74–76]. Nevertheless, the methods used here are briefly described for completeness.

The features used here are the centroids of the object and that of the table. The first step is to segment these from the image. Color segmentation is used in our system. The ball has a known color, and it can be easily characterized and thresholded by a specific hue range and a high saturation (from the HSV color space). To add robustness, morphological operations (opening and closing) are used to remove outliers. After this, sliding window detection (sped up using the image pyramids concept) finds the most probable location of the ball. The centroid of the detected blob (corresponding to the ball) is denoted with (u, v) in pixel coordinates. This is then converted into ${}^{c}x_o$ and ${}^{c}y_o$ by using the intrinsic camera calibration parameters (f_x, f_y, c_x, c_y) and the depth ${}^{c}z_o$ as follows:

$$ {}^{c}x_o = \frac{{}^{c}z_o(u-c_x)}{f_x} \quad , \quad {}^{c}y_o = \frac{{}^{c}z_o(v-c_y)}{f_y}. \tag{2.7} $$

The next step is to segment the table in the image. A flood fill algorithm [76] is run in saturation-value-depth space. This algorithm starts with a *seed* point and grows the region based on a connectivity criterion between neighboring pixels. Here, the seed point is the bottom pixel of the ball. A low saturation and high value characterize well the *white* color of the table. The addition of depth ensures connectivity in Cartesian space, simplifying for example the segmentation between table and floor pixels. Finally, some morphological operations (opening and closing) are done to remove outliers. From these segmented points, the Cartesian centroid is used as ${}^{c}\mathbf{t}_t$ (a translation vector). The Cartesian coordinates of the object in the table frame

are then obtained by applying:

$$ {}^{t}\mathbf{t}_o = {}^{c}\mathbf{T}_t^{-1}\ {}^{c}\mathbf{t}_o. \tag{2.8} $$

The homogeneous transformation matrix ${}^{c}\mathbf{T}_t$ is composed of the table centroid position ${}^{c}\mathbf{t}_t$ and the rotation matrix ${}^{c}\mathbf{R}_t$. A simple approximation consists in setting ${}^{c}\mathbf{R}_t$ equal to ${}^{c}\mathbf{R}_{cf}$, which is obtained from proprioception.

The control design needs to drive ${}^{t}\mathbf{t}_o$ to ${}^{t}\mathbf{t}_d$. Several existing methods can be used. Here, a simple PD controller is used such that:

$$ C_i(s) = K_{p,i} + K_{d,i}s \qquad i = \{x, y\}\,. \tag{2.9} $$

This choice is justified by analyzing the task using a simple sliding model (i.e., neglecting friction and angular momentum). Figure 2.3 illustrates the necessary variables for this analysis. Since a control with z rather than ϕ_y is desired, the trigonometric identity $z_r = l_t \sin\phi_y$ is used, where l_t is the length of the table and z_r is the differential height. z_r can be converted to z by a trivial change of frame.

The Lagrangian equation of motion along ${}^{t}\vec{x}$ is:

$$ m\ddot{x} = mg\sin\phi_y = mgz_r/l_t. \tag{2.10} $$

Along y, linearization of the Lagrangian equation about $\phi_x = 0$ leads to:

$$ m\ddot{y} = -mg\phi_x. \tag{2.11} $$

Taking the Laplace transforms of these two equations yields:

$$ \begin{cases} s^2X(s) = gZ_r\,(s)\,/l_t \\ s^2Y(s) = -g\Phi\,(s)\,. \end{cases} \tag{2.12} $$

Rearranging, the transfer functions describing the dynamics of the 2 DOF can be derived:

$$ \begin{cases} P_x(s) = \frac{X(s)}{Z_r(s)} = \frac{g}{l_t s^2} \\ P_y(s) = \frac{Y(s)}{\Phi(s)} = -\frac{g}{s^2}. \end{cases} \tag{2.13} $$

It should be noted that both are double integrators. As such, they are only marginally stable when feedback controlled with a Proportional gain. But a Proportional Derivative controller (PD) can be used. The denominator of the closed loop system transfer function in the two cases is:

$$ \begin{cases} D_x(s) = l_t s^2 + gK_{d,x}s + gK_{p,x} \\ D_y(s) = s^2 - gK_{d,y}s - gK_{p,y}. \end{cases} \tag{2.14} $$

The two systems are asymptotically stable if all the roots of these two polynomials have non-multiple negative real parts. This condition is verified, for a second order

polynomial, if all the coefficients are strictly positive. In the case of the characteristic polynomials in (2.14), this is equivalent to:

$$K_{p,x} > 0 \qquad K_{d,x} > 0 \qquad K_{p,y} < 0 \qquad K_{d,y} < 0. \tag{2.15}$$

Finally, the applied controllers are:

$$\begin{cases} z = K_{p,x}\left(x_d - x\right) - K_{d,x}\dot{x} \\ \phi_x = K_{p,y}\left(y_d - y\right) - K_{d,y}\dot{y}. \end{cases} \tag{2.16}$$

By numerical differentiation $\dot{x}$ (and $\dot{y}$) is obtained as:

$$\dot{x}\left(t\right) = \frac{x\left(t\right) - x\left(t - \Delta t\right)}{\Delta t},$$

with Δt the sampling step. Tuning the gains in (2.16) according to (2.15) guarantees stability of the closed loop system, as long as the linear approximation is valid. This implies that ${}^t\mathbf{t}_o$ will converge to ${}^t\mathbf{t}_d$, as desired. The outputs of (2.16) are fed to the admittance controller (2.1) as desired values z_d and $\phi_{x,d}$. Numerical differentiation is used to obtain $\dot{z}, \dot{\phi}_x$ in $\dot{\mathbf{p}}_d$. However, for $\ddot{\mathbf{p}}_d$ a piece-wise constant velocity is assumed, such that $\ddot{z} = \ddot{\phi}_x = 0$. This also prevents too much noise to be introduced by a second numerical differentiation. The results of this are presented in section 2.3.

2.3 Results and discussion

The case studies of section 2.2 were tested experimentally on the HRP-2 humanoid robot from Kawada Industries. These were also integrated with previous works in our research group [10, 49], that realize the collaborative transportation task by acting on the x, y, ϕ_z DOF. Each of the cases is presented in the corresponding subsections that follow. A discussion about these cases in the context of human-humanoid collaboration follows in subsection 2.3.3.

2.3.1 Stationary Object - keeping the plane level

Several experiments were performed. The first is the simplest, involving just the arms of the robot standing in place. Next, the HRP-2 is made to walk in place, introducing the perturbations from walking. Finally, a full experiment is done, to show that the work here integrates well to the previous works [10, 49]. Figure 2.7 shows some qualitative results taken during the task. The top pictures show the first test (standing in place). The bottom pictures show the third test, with the robot walking together with the human.

Some early experiments show that holding the table too high while walking can cause the robot to fall down because of the human's pulling force. This can be explained by the fact that more torque is applied on the humanoid, since the lever

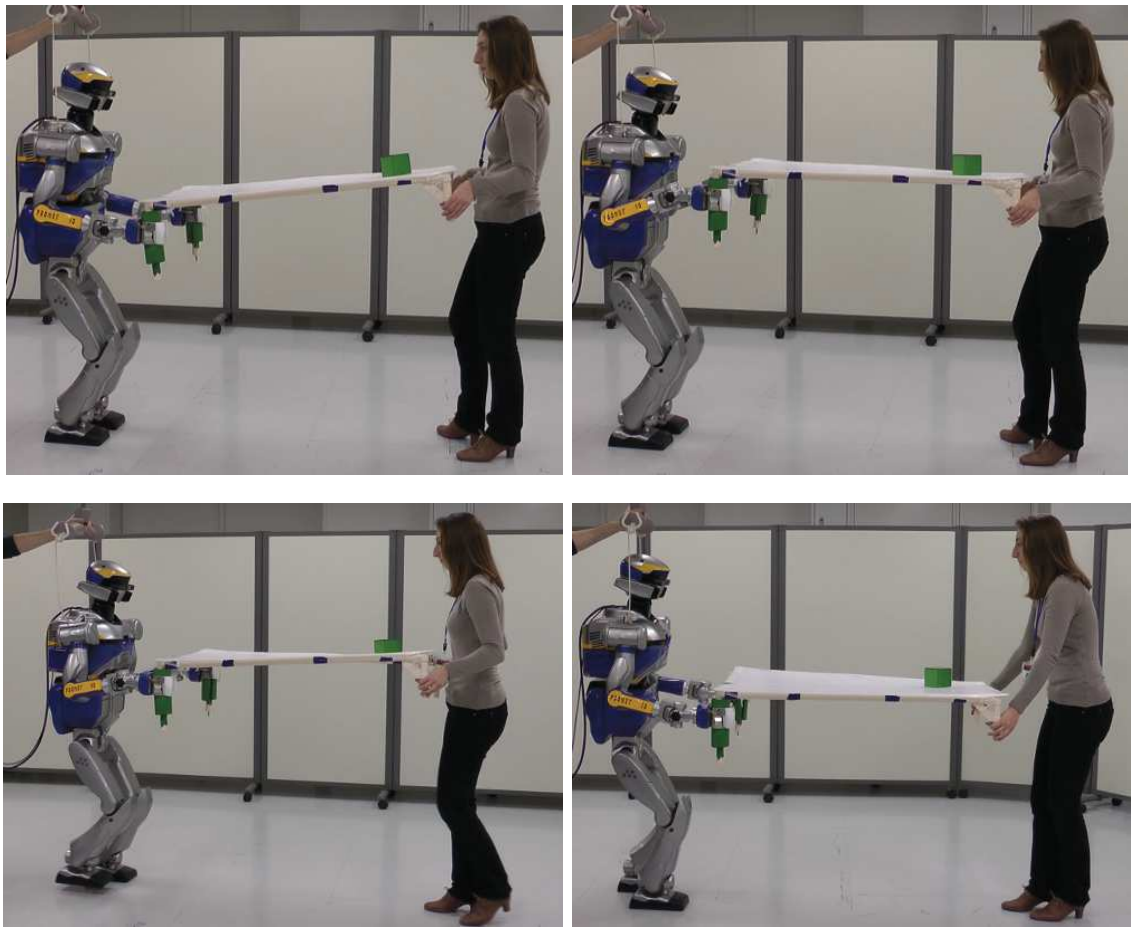


Figure 2.7: Snapshots of an experiment for the stationary object (green cube): top left - before vision control is activated, top right - result after the vision controller is activated, bottom left and right - results while walking with the robot and changing table height.

arm is increased when the table is held high. So the experiments shown here have been realized with a saturation limit on the Z motion to prevent this. This is however only a temporary fix, as the main issue is handled directly in chapter 3.

To verify the controller design, a step response of the error (ϕ_y) is obtained from an experiment where the human holds his/her end steady with $Z_r \neq 0$ at the beginning (Fig. 2.7, top left). The controller is then activated and the robot corrects the height difference, ending at Fig. 2.7, top right. Angle ϕ_y for this sequence is plotted in Fig. 2.8, (left). This result shows an exponential decrease (although with some noise, and notable latency of the vision algorithm), implying that the implementation follows the design.

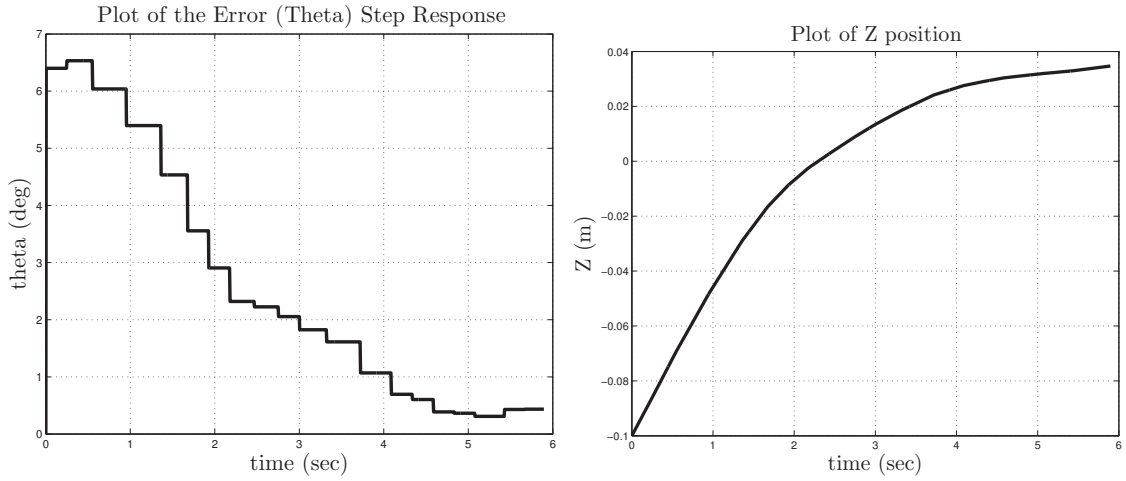


Figure 2.8: Step response of the vision controller. Top: Plot of the error (ϕ_y). Bottom: Plot of ${}^{l}z_{cf}$

2.3.2 Moving Object - keeping a ball from falling off

For the experiments of the second case study, we chose a ball to be the moving object. This makes it similar to a well-studied problem/example in control theory: the *ball-and-plate* system, which is a 2-DOF generalization of the *textbook example* ball-on-beam system (used to study advanced control methods [82]). Although similar, our context for using it is different: notably we are focused on the human-robot collaboration aspect, together with the combination of vision and haptics.

Several experiments were performed and with 2 different balls - a yellow tennis ball, which tends to move slower, and a pink ball, which moves quite fast. A few different users also tested this early system, but as the described experience was similar this is not discussed here. Some snapshots of the experiments are shown in Fig. 2.9. In the initial experiments, both human and humanoid stand stationary and balance the ball on the table. Some disturbance is then introduced (e.g. the ball is pushed by another person) and the gains of the PD controller are tuned, according to (2.15), in order to be able to handle such a disturbance. After some gain tuning of the vision-based controller with such experiments, we test the complete system where the human-humanoid dyad transport the table with the ball on top. Here, walking introduces a significant disturbance, that can move the ball. The experiments show that, although the ball moves a lot, it doesn't fall off the table during this transportation task.

From the recorded data of the force/torque sensors in the robot wrists, we found that during this task τ_x (the total torque about the x-axis of {cf}) averages to about 0 Nm, which means that this interaction torque with the human is regulated well. Furthermore, f_z (again with {cf} as the reference) averages to about 12 N. This means that the robot carries part of the weight of the table and thus lightens the burden on the human. Finally, we notice that in both signals a noticeable oscillation

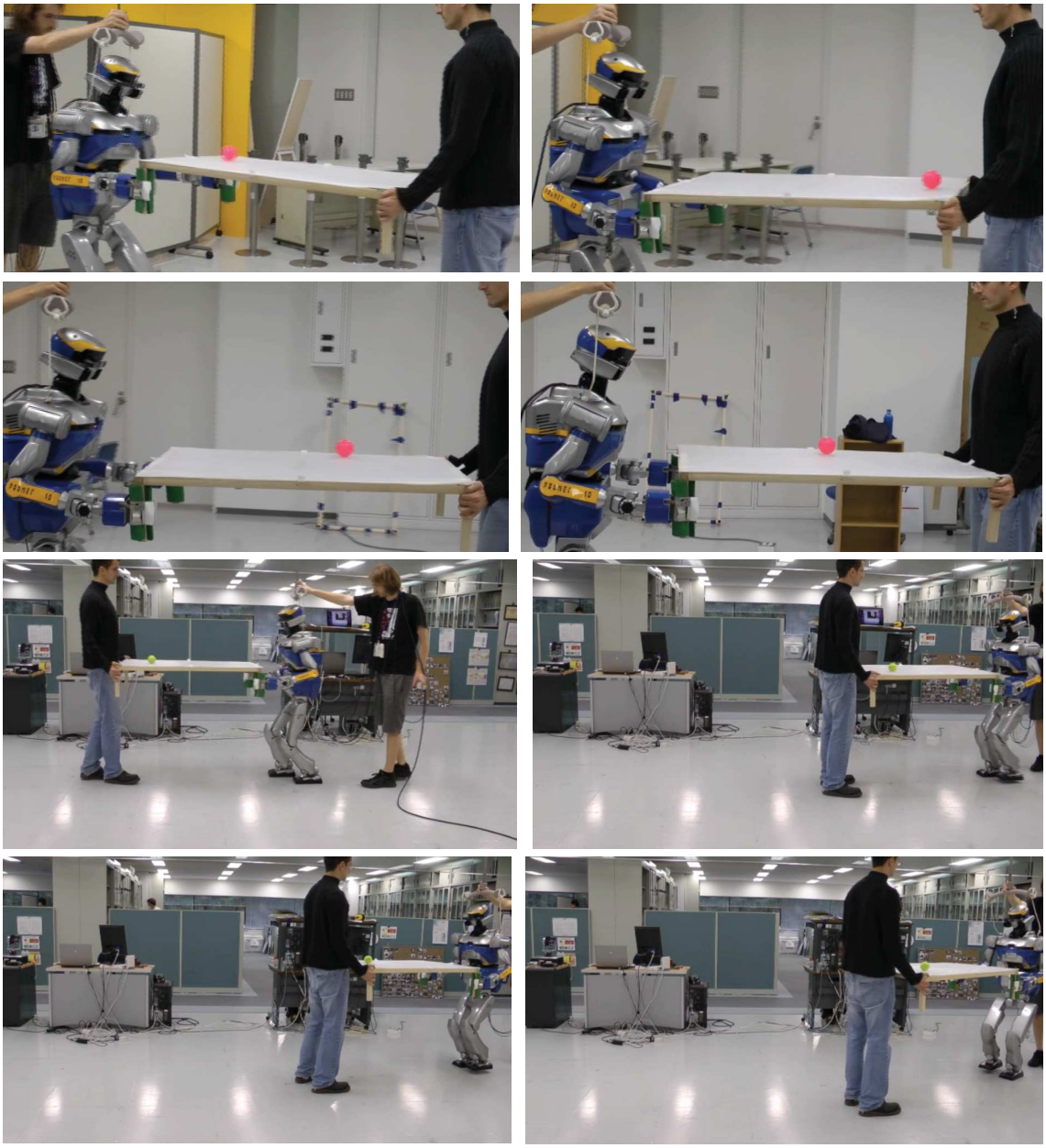


Figure 2.9: Snapshots of two experiments, where the human-humanoid dyad transports a table with a freely-moving ball on top (a fast moving ball in the top sequence, and a slow moving ball in the bottom one).

occurs. This correlates to the frequency of the walking gait, hence to the disturbance that it causes.

2.3.3 Discussion

The results show that the framework presented can do the job well: the vision-based controller tries to do its task (keep the table level in the case with the cube, or keep the ball on the table in the second case) while the impedance controller regulates interaction forces. Relating it back to our previous works, we can see now that, for the pure follower (depicted in Fig. 2.1 a), the success/failure of the vision task depends solely on the human partner. Specifically, the human needs to use his/her vision to observe the state of the task and then apply a sufficient force to haptically communicate to the robot what s/he wants to do. Instead, in both of our case studies (depicted by Fig. 2.1 c), the cognitive load of the task is shared in some capacity - both human and robot are able to observe the state of the task and to act accordingly. However, this sharing can become a disadvantage when the human and robot disagree on the state of the task and on the action to take [3]. Experimentally, this is handled in our system, by making the robot more compliant and less stiff (impedance parameter tuning). This ensures that the human can always safely impose his/her intention through the haptic channel. This also shows a possible extension of the system which consists in dynamically changing the impedance parameters, to make it more stiff when the robot is more certain of its observations and more compliant when there is more uncertainty. In effect, this makes the impedance parameters a tool to weigh the importance between visual (task knowledge) and haptic (human intention) information channels. However, it is important to note that this disadvantage of equal collaboration also applies to human-human pairs and more generally to teams - *teamwork* (or the lack of it). Our experiments have been made with both the passive follower, and with the approach of equal collaboration, and the advantages/disadvantages briefly described here can be observed by the human collaborator. One difficulty in presenting these results is in the use of proper evaluation methods, since the most important aspect - the comfort of the human collaborator - is very subjective. Another difficulty is to separate the contribution of the human and robot. Although in the results presented here the human is told to be more *passive* (for example: does not try *too much* to keep the ball on the table) he also does not try to make the ball fall off, since teamwork is a factor in the overall result.

The tests in this chapter allowed us to gain insight into some of the possibilities for adding vision into haptic joint actions. However, it also exposed some flaws in other areas that needed work. First, we briefly mentioned that we had to limit the z movements and interaction. The reason is that in this work, we directly used the walking pattern generator of [47], which is not specifically designed for walking while physically interacting. Chapter 3 will show why this walking pattern generator leads to the failures we have seen here, and how to improve on it. Moreover, in this chapter, the hand tasks are realized by relying on the stack-of-tasks [33] inverse kinematics solver. In chapter 4, we go deeper into the details of whole-body control, to better integrate all the tasks required for physical human-robot interaction.

Chapter 3

Walking designed for physical collaboration

To be effective in physical collaboration, a humanoid robot must locomote in the environment, in addition to having manipulation capabilities. This chapter focuses on locomotion, specifically on walking in physical collaboration scenarios.

In the robotics literature, walking has historically been treated separately from manipulation. This has resulted in a good understanding of the underlying principles behind walking and locomotion in general. Although this is a good starting point, both manipulation and locomotion need to be consistent with each other, particularly when physical collaboration is necessary. Eventually, both need to be thought of as parts of the whole-body control problem (to be discussed in the next chapter). In this chapter, we start by revisiting the modeling of walking pattern generators, and eventually redesign these, with physical collaboration in mind.

A robot that locomotes in its environment can be described as having a *floating-base* [30], that defines the pose of the robot base in a global reference frame. This floating-base extends the configuration space of the robot and makes it different from *fixed-base* robots (e.g., industrial manipulators, that have their own workspace). The configuration space can then be written as:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}' \\ \mathbf{p}_{\text{fl}} \end{bmatrix}, \tag{3.1}$$

where $\mathbf{q}'$ describes the robot joint positions, and $\mathbf{p}_{\text{fl}}$ the pose of the floating base. The simple goal of locomotion is to be able to control $\mathbf{p}_{\text{fl}}$ to some extent. This is not straightforward, as can be seen by inspecting the dynamic equations:

$$\mathbf{H}_{\text{fl}}\ddot{\mathbf{q}} + \mathbf{C}_{\text{fl}} = \begin{bmatrix} \boldsymbol{\tau} \\ \mathbf{0} \end{bmatrix} + \sum_i \mathbf{J}_{\text{i}}^{\top} \mathbf{f}_{\text{i}}, \tag{3.2}$$

where $\mathbf{H}_{\text{fl}}$ and $\mathbf{C}_{\text{fl}}$ are respectively the inertia and Coriolis/centrifugal terms, extended to the floating base [30], $\mathbf{J}_i$ is the Jacobian matrix at the i-th contact point,

$\mathbf{f}_i$ are the contact forces, and $\boldsymbol{\tau}$ the joint torques. The locomotion problem can be seen in the right hand side of Eq.(3.2): the floating-base does not have direct torque actuation, as indicated by the zero vector. If it did, locomotion would become simpler. Instead, in order to effectively control the floating base, we need to use the contact forces with the environment. Another consequence of model (3.2), is that the part of the dynamics that is not directly actuated still depends on the Newton-Euler equations of the floating base (see [38], with more details in [39]). This means that the essence of locomotion and balance will consist in generating Center of Mass (CoM) motions that are consistent with the dynamics and contact states (i.e., footsteps). We will take advantage of these considerations, instead of tackling Eq.(3.2) directly.

This chapter starts by revisiting the robot dynamic model, to account for the physical interaction between robot and human. This new model is then used to generate the walk, by expressing it as an optimization problem, in the framework of model predictive control. Finally, the optimization problem is reformulated specifically for physical collaboration, and validated through simulations and real robot experiments.

3.1 Dynamic walk model

Before anything else, a choice must be made on how to formulate the reduced dynamic model of the robot, in order to consider physical interaction. Three possibilities for this are proposed in [83]. The differences can be thought of as moving the abstraction layer of the physical interaction. The reduced models proposed in [83] are:

1. a model with full knowledge of the object (and/or human),
2. a model that considers the effects of the object (and/or human) on the robot's contact locations and linear forces, requiring additional grasp stability constraints,
3. a model that considers the effects of the object (and/or human) as external wrenches applied on the robot.

In [83], the first option was chosen and demonstrated in a simulation of HRP-2 carrying a box. The model showed good results in simulation, and can be implemented on a real robot, by using the hand force sensors to estimate the object's dynamics beforehand. If this model is to be used in the context of collaborative carrying, it must consider the robot, human and object as a single system. The CoM of this system has contributions from the three subsystems. Also, there can be four (2 for the robot, 2 for the human) possible foot contacts - as for a *quadruped*. This model can be used in simulation to control only the robot, while having perfect knowledge of each subsystem (human, robot, object). However, it clearly requires important

resources in terms of sensing, making it hardly usable in practice. The second option corresponds to a non-linear model [83]. We have chosen to avoid this complexity. This leaves us with the third option, which is chosen because of its simplicity in terms of implementation on a real robot, considering its application to human-robot interaction, in general, and to collaborative carrying, in particular.

The development of this reduced model is inspired by [38]. However, the particularity of our model, relative to [38], is to explicitly separate the foot contact forces with the plane $\mathbf{f}_{\mathrm{i}}$, from all other contact forces with the environment, which are represented as an *external wrench* in the CoM frame: $\mathbf{h}_{\mathrm{ext}} = [\mathbf{f}_{\mathrm{ext}}^{\top}\ \mathbf{n}_{\mathrm{ext}}^{\top}]^{\top}$. The external wrench is separated because it will go on to represent physical interaction (e.g., with a human or in pushing or carrying objects). The resulting Newton and Euler equations of motion are now:

$$m(\ddot{\mathbf{c}} + \mathbf{g}) = \mathbf{f}_{\mathrm{ext}} + \sum_i \mathbf{f}_{\mathrm{i}} \tag{3.3}$$

$$\dot{\mathbf{L}} = \mathbf{n}_{\mathrm{ext}} + \sum_i (\mathbf{r}_{\mathrm{i}} - \mathbf{c}) \times \mathbf{f}_{\mathrm{i}}, \tag{3.4}$$

where m is the mass of the robot, $\mathbf{c}$ symbolizes the CoM position in a fixed inertial reference frame, $\mathbf{g}$ the acceleration due to gravity, $\mathbf{L}$ the angular momentum and $\mathbf{r}_{\mathrm{i}}$ the position of the foot contact points (also in the inertial frame). Pre-multiplying the first equation by $\mathbf{c}$, and adding to the second, we obtain:

$$\sum_i \mathbf{r}_{\mathrm{i}} \times \mathbf{f}_{\mathrm{i}} = m\mathbf{c} \times (\ddot{\mathbf{c}} + \mathbf{g}) + \dot{\mathbf{L}} - \mathbf{n}_{\mathrm{ext}} - (\mathbf{c} \times \mathbf{f}_{\mathrm{ext}}). \tag{3.5}$$

Dividing Eq.(3.5) by the z component of the feet contact forces while using Eq.(3.3), yields:

$$\frac{\sum_i \mathbf{r}_{\mathrm{i}} \times \mathbf{f}_{\mathrm{i}}}{\sum_i f_{\mathrm{i}}^{\mathrm{z}}} = \frac{m\mathbf{c} \times (\ddot{\mathbf{c}} + \mathbf{g}) + \dot{\mathbf{L}} - \mathbf{n}_{\mathrm{ext}} - (\mathbf{c} \times \mathbf{f}_{\mathrm{ext}})}{m(\ddot{c}^{\mathrm{z}} + g^{\mathrm{z}}) - f_{\mathrm{ext}}^{\mathrm{z}}}. \tag{3.6}$$

We assume that the robot is walking on a flat ground, with all contact points between the feet and the ground having the same z coordinate, $\mathbf{r}_{\mathrm{i}}^{\mathrm{z}} = 0$, and with a constant height above the ground, so $\ddot{c}^{\mathrm{z}} = 0$. This leads to a linear relationship between the position of the CoM and the position of the Center of Pressure. For the sake of simplicity, we also assume here that variations of the angular momentum are negligible, so $\dot{\mathbf{L}} = \mathbf{0}$, and that gravity is orthogonal to the ground, thus the constant $\mathbf{g}^{\mathrm{x,y}} = \mathbf{0}$. With these assumptions, we can then simplify the x and y components of (3.6) to get:

$$\frac{\sum_i f_{\mathrm{i}}^{\mathrm{z}} \mathbf{r}_{\mathrm{i}}^{\mathrm{x,y}}}{\sum_i f_{\mathrm{i}}^{\mathrm{z}}} = \left(\frac{mg^{\mathrm{z}}}{mg^{\mathrm{z}} - f_{\mathrm{ext}}^{\mathrm{z}}}\right)\left(\mathbf{c}^{\mathrm{x,y}} - \frac{c^{\mathrm{z}}}{g^{\mathrm{z}}}\ddot{\mathbf{c}}^{\mathrm{x,y}}\right) - \mathbf{R}_{\mathrm{mod}}\left(\frac{\mathbf{n}_{\mathrm{ext}}^{\mathrm{x,y}} + (\mathbf{c} \times \mathbf{f}_{\mathrm{ext}})^{\mathrm{x,y}}}{mg^{\mathrm{z}} - f_{\mathrm{ext}}^{\mathrm{z}}}\right), \tag{3.7}$$

where $\mathbf{R}_{\mathrm{mod}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

Before moving on with the modeling, it is important to explain the meaning of Eq.(3.7). Firstly, note that the left hand side of the equation is the definition of the Center of Pressure (CoP). Obtaining this was the purpose of all the algebraic manipulations so far. Because the ground reaction forces are unilateral (strictly positive), the CoP must belong to the convex hull of the contact points, $\mathbf{r}_{\mathrm{i}}$. Note that the CoP is also known as the Zero Moment Point (ZMP) in the literature, and that this term will be used from here on. Furthermore, the convex hull of the feet contact points is often referred to as the *support polygon*. The goal of most walking algorithms is in fact to keep the ZMP within the support polygon [37], which enforces consistency with the contact dynamics.

Further simplifying and grouping terms, we end up with the following expression of the ZMP denoted by $\mathbf{z}^{\mathrm{x,y}}$:

$$\mathbf{z}^{\mathrm{x,y}} = \mathbf{c}^{\mathrm{x,y}} - \left(\frac{c^{\mathrm{z}}}{g^{\mathrm{z}} - \frac{f^{\mathrm{z}}_{\mathrm{ext}}}{m}}\right)\ddot{\mathbf{c}}^{\mathrm{x,y}} - \mathbf{R}_{\mathrm{mod}}\left(\frac{\mathbf{n}^{\mathrm{x,y}}_{\mathrm{ext}}}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}}\right) + \left(\frac{c^{\mathrm{z}}\mathbf{f}^{\mathrm{x,y}}_{\mathrm{ext}}}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}}\right). \tag{3.8}$$

When there is no external wrench, this expression simplifies to

$$\mathbf{z}^{\mathrm{x,y}} = \mathbf{c}^{\mathrm{x,y}} - \left(\frac{c^{\mathrm{z}}}{g^{\mathrm{z}}}\right)\ddot{\mathbf{c}}^{\mathrm{x,y}}, \tag{3.9}$$

which is the standard expression found in the literature for the ZMP. Guidelines on how to reduce the effects of an external wrench can be inferred from the structure of equation (3.8). Typically: higher robot mass, lower CoM height, and weak external force components along x and y, would reduce the external wrench effects on the ZMP position.

To generate smooth motions of the CoM, we assume that its trajectory is differentiable three times. This allows us to choose the CoM jerk as control input. Concatenating the x and y DOF, we can define:

$$\hat{\mathbf{c}} = \begin{bmatrix} c^{\mathrm{x}} \\ \dot{c}^{\mathrm{x}} \\ \ddot{c}^{\mathrm{x}} \\ c^{\mathrm{y}} \\ \dot{c}^{\mathrm{y}} \\ \ddot{c}^{\mathrm{y}} \end{bmatrix}, \quad \dddot{\mathbf{c}}^{\mathrm{x,y}} = \begin{bmatrix} \dddot{c}^{\mathrm{x}} \\ \dddot{c}^{\mathrm{y}} \end{bmatrix}, \quad \mathbf{z}^{\mathrm{x,y}} = \begin{bmatrix} z^{\mathrm{x}} \\ z^{\mathrm{y}} \end{bmatrix}, \quad \hat{\mathbf{f}} = \begin{bmatrix} n^{\mathrm{y}}_{\mathrm{ext}} \\ f^{\mathrm{x}}_{\mathrm{ext}} \\ n^{\mathrm{x}}_{\mathrm{ext}} \\ f^{\mathrm{y}}_{\mathrm{ext}} \end{bmatrix}, \tag{3.10}$$

such that the continuous time linear model is:

$$\begin{aligned} \dot{\hat{\mathbf{c}}} &= \mathbf{A}\hat{\mathbf{c}} + \mathbf{B}\dddot{\mathbf{c}}^{\mathrm{x,y}}, \\ \mathbf{z}^{\mathrm{x,y}} &= \mathbf{D}\hat{\mathbf{c}} + \mathbf{E}\hat{\mathbf{f}}. \end{aligned} \tag{3.11}$$

where:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & -\frac{mc^{\mathrm{z}}}{(mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}})} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{mc^{\mathrm{z}}}{(mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}})} \end{bmatrix},$$

$$\mathbf{E} = \begin{bmatrix} \frac{1}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} & \frac{c^{\mathrm{z}}}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} & 0 & 0 \\ 0 & 0 & -\frac{1}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} & \frac{c^{\mathrm{z}}}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} \end{bmatrix}.$$

Discretization leads to:

$$\begin{aligned} \hat{\mathbf{c}}_{\mathrm{k+1}} =& \mathbf{A}_{\mathrm{k}} \hat{\mathbf{c}}_{\mathrm{k}} + \mathbf{B}_{\mathrm{k}} \dddot{\mathbf{c}}^{\mathrm{x,y}}_{\mathrm{k}}, \\ \mathbf{z}^{\mathrm{x,y}}_{\mathrm{k+1}} =& \mathbf{D}_{\mathrm{k+1}} \hat{\mathbf{c}}_{\mathrm{k+1}} + \mathbf{E}_{\mathrm{k+1}} \hat{\mathbf{f}}_{\mathrm{k+1}} \\ =& \mathbf{D}_{\mathrm{k+1}} \mathbf{A}_{\mathrm{k}} \hat{\mathbf{c}}_{\mathrm{k}} + \mathbf{D}_{\mathrm{k+1}} \mathbf{B}_{\mathrm{k}} \dddot{\mathbf{c}}^{\mathrm{x,y}}_{\mathrm{k}} + \mathbf{E}_{\mathrm{k+1}} \hat{\mathbf{f}}_{\mathrm{k+1}}, \end{aligned} \tag{3.12}$$

where:

$$\mathbf{A}_{\mathrm{k}} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B}_{\mathrm{k}} = \begin{bmatrix} \frac{\Delta t^3}{6} & 0 \\ \frac{\Delta t^2}{2} & 0 \\ \Delta t & 0 \\ 0 & \frac{\Delta t^3}{6} \\ 0 & \frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix},$$

$$\mathbf{D}_{\mathrm{k}} = \begin{bmatrix} 1 & 0 & -\frac{mc^{\mathrm{z}}}{(mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}})} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{mc^{\mathrm{z}}}{(mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}})} \end{bmatrix},$$

$$\mathbf{E}_{\mathrm{k}} = \begin{bmatrix} \frac{1}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} & \frac{c^{\mathrm{z}}}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} & 0 & 0 \\ 0 & 0 & -\frac{1}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} & \frac{c^{\mathrm{z}}}{mg^{\mathrm{z}} - f^{\mathrm{z}}_{\mathrm{ext}}} \end{bmatrix},$$

with Δt the sampling period, and the subscript k or $k+1$ denoting the discrete time interval.

In the following Section, we will explain how Model Predictive Control will be used to regulate the dynamics of the CoM expressed by Eq.(3.12). Before formulating this controller, there are some things to note on the matrices $\mathbf{A}_{\mathrm{k}}, \mathbf{B}_{\mathrm{k}}, \mathbf{D}_{\mathrm{k}}, \mathbf{E}_{\mathrm{k}}$. First, it is possible to vary the individual variables from one time interval to the next. By keeping all the variables constant, the model becomes a Linear Time-Invariant (LTI) system. Allowing some or all variables to vary results in a Linear Time-Variant (LTV) system. In particular, varying c^{z} and $f^{\mathrm{z}}_{\mathrm{ext}}$ may be interesting. In this work, we assume for simplicity that c^{z} and $f^{\mathrm{z}}_{\mathrm{ext}}$ are constant, leading to an LTI system. These are reasonable assumptions: the first is common in the walking

literature, and the second is true if the mass of the transported object does not vary over time. The LTI formulation has the advantage of leading to some simplifications which eventually reduce computation time. However, it is possible to extend to the LTV formulation for Model Predictive Control: the main issue will be in knowing/predicting the variation in advance. The second thing to note is that the effect of n^z_{ext} has been neglected implicitly. This was a result of the assumptions in the ZMP derivation to keep the formulation linear. Similarly, [83] notes that the contribution of L^z should be ignored to keep the formulation linear. Keeping these aspects in mind, the model representation of Eq.(3.12) can now be used to formulate the walking pattern generator.

3.2 Walking pattern generator

To generate a walking pattern that accounts for external forces, we first express the dynamic walk model in the framework of model predictive control (Sect. 3.2.1). Then, within this framework, we express it as an optimization problem (Sect. 3.2.2). Finally, we explain how the swing foot trajectories are designed, in order to implement the walk (Sect. 3.2.3).

3.2.1 Model Predictive Control

Model Predictive Control (MPC) is a method for controlling a system, so that future states are also taken into account. This makes it effective in walking motions, as demonstrated in [41, 47], since the generation of a smooth CoM motion, requires the anticipation of the foot contact changes (future footsteps). Otherwise, motion is only generated just after the current footstep, to recover from the instantaneous contact transition.

A common MPC methodology consists in eliminating future states by iteratively applying the model over N discrete steps (termed the *preview or prediction horizon*). This results in a new problem formulation where the previewed future states are a function of only the current state and of the current and future control inputs. Doing so for (3.12) results in:

$$\begin{aligned} \tilde{\mathbf{c}} &= \mathbf{U}_{\text{c}}\hat{\mathbf{c}}_0 + \mathbf{U}_{\text{u}}\tilde{\mathbf{u}}, \\ \tilde{\mathbf{z}} &= \mathbf{O}_{\text{c}}\hat{\mathbf{c}}_0 + \mathbf{O}_{\text{u}}\tilde{\mathbf{u}} + \mathbf{O}_{\text{f}}\tilde{\mathbf{f}}, \end{aligned} \tag{3.13}$$

where:

$$\tilde{\mathbf{c}} = \begin{bmatrix} \hat{\mathbf{c}}_1 \\ \vdots \\ \hat{\mathbf{c}}_{\text{N}} \end{bmatrix}, \quad \tilde{\mathbf{u}} = \begin{bmatrix} \dddot{\mathbf{c}}^{\text{x,y}}_0 \\ \vdots \\ \dddot{\mathbf{c}}^{\text{x,y}}_{\text{N-1}} \end{bmatrix}, \quad \tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{z}^{\text{x,y}}_1 \\ \vdots \\ \mathbf{z}^{\text{x,y}}_{\text{N}} \end{bmatrix}, \quad \tilde{\mathbf{f}} = \begin{bmatrix} \hat{\mathbf{f}}_1 \\ \vdots \\ \hat{\mathbf{f}}_{\text{N}} \end{bmatrix},$$

are the concatenation of states, controls, outputs and external forces respectively, in

the preview horizon of length N. The matrices:

$$\mathbf{U}_\mathrm{c} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1\mathbf{A}_0 \\ \vdots \\ \mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_0 \end{bmatrix}, \mathbf{U}_\mathrm{u} = \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_1\mathbf{B}_0 & \mathbf{B}_1 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_1\mathbf{B}_0 & \mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_2\mathbf{B}_1 & \dots & \mathbf{B}_{\mathrm{N\text{-}1}} \end{bmatrix},$$

$$\mathbf{O}_\mathrm{c} = \begin{bmatrix} \mathbf{D}_1\mathbf{A}_0 \\ \mathbf{D}_2\mathbf{A}_1\mathbf{A}_0 \\ \vdots \\ \mathbf{D}_\mathrm{N}\mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_0 \end{bmatrix}, \mathbf{O}_\mathrm{f} = \begin{bmatrix} \mathbf{E}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{E}_\mathrm{N} \end{bmatrix},$$

$$\mathbf{O}_\mathrm{u} = \begin{bmatrix} \mathbf{D}_1\mathbf{B}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{D}_2\mathbf{A}_1\mathbf{B}_0 & \mathbf{D}_2\mathbf{B}_1 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_\mathrm{N}\mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_1\mathbf{B}_0 & \mathbf{D}_\mathrm{N}\mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_2\mathbf{B}_1 & \dots & \mathbf{D}_\mathrm{N}\mathbf{B}_{\mathrm{N\text{-}1}} \end{bmatrix},$$

complete the description and are obtained by the state elimination/*condensing* process which is detailed in Appendix A. From the second equation of (3.13), it will be useful to decompose the global ZMP positions $\tilde{\mathbf{z}}$. For a single instance ${}^{\mathrm{w}}\mathbf{z}$, this results in:

$$\begin{aligned} {}^{\mathrm{w}}\mathbf{z} &= {}^{\mathrm{w}}\mathbf{T}_{\mathbf{r}_0}{}^{\mathbf{r}_0}\mathbf{T}_{\mathbf{r}_\mathrm{j}}{}^{\mathbf{r}_\mathrm{j}}\mathbf{z} \\ &= {}^{\mathrm{w}}\mathbf{r}_0 + {}^{\mathrm{w}}\mathbf{R}_{\mathbf{r}_0}{}^{\mathbf{r}_0}\mathbf{r}_\mathrm{j} + {}^{\mathrm{w}}\mathbf{R}_{\mathbf{r}_\mathrm{j}}{}^{\mathbf{r}_\mathrm{j}}\mathbf{z} \\ &= {}^{\mathrm{w}}\mathbf{r}_0 + \sum_{i=1}^{M} {}^{\mathrm{w}}\mathbf{R}_{\mathbf{r}_{\mathrm{i\text{-}1}}}{}^{\mathbf{r}_{\mathrm{i\text{-}1}}}\mathbf{r}_\mathrm{j} + {}^{\mathrm{w}}\mathbf{R}_{\mathbf{r}_\mathrm{j}}{}^{\mathbf{r}_\mathrm{j}}\mathbf{z}. \end{aligned} \tag{3.14}$$

This decomposition has two purposes. Firstly, it expresses the ZMP in a frame relative to its corresponding footstep ${}^{\mathbf{r}_\mathrm{j}}\mathbf{z}$. This allows us to give it simple bounds, later in the constraint formulation. Secondly, it provides an expression of the footsteps relative to the previous ones ${}^{\mathbf{r}_{\mathrm{i\text{-}1}}}\mathbf{r}_\mathrm{j}$, $i = 1, \dots, M$. This representation further allows the controller to modify the footsteps online, as in [47]. Extending over the N steps of the preview horizon, we can write:

$$\tilde{\mathbf{z}} = \mathbf{v}_{\mathbf{r}_0} + \mathbf{V}_\mathrm{r}\breve{\mathbf{r}} + \mathbf{V}_\mathrm{z}\breve{\mathbf{z}}, \tag{3.15}$$

where:

$$\breve{\mathbf{z}} = \begin{bmatrix} {}^{\mathbf{r}_1}\mathbf{z}_1^{\mathrm{x,y}} \\ \vdots \\ {}^{\mathbf{r}_\mathrm{N}}\mathbf{z}_\mathrm{N}^{\mathrm{x,y}} \end{bmatrix}, \quad \breve{\mathbf{r}} = \begin{bmatrix} {}^{\mathbf{r}_0}\mathbf{r}_1^{\mathrm{x,y}} \\ \vdots \\ {}^{\mathbf{r}_{\mathrm{M\text{-}1}}}\mathbf{r}_\mathrm{M}^{\mathrm{x,y}} \end{bmatrix}, \quad \mathbf{v}_{\mathbf{r}_0} = \begin{bmatrix} {}\mathbf{r}_0{}^{\mathrm{x,y}} \\ \vdots \\ {}\mathbf{r}_0{}^{\mathrm{x,y}} \end{bmatrix}, \tag{3.16}$$

are respectively the local ZMP position, local variable footstep positions and the last known global frame footstep positions. Note that $\breve{\mathbf{r}} \in \mathbb{R}^{2\mathrm{M}}$, while $\breve{\mathbf{z}} \in \mathbb{R}^{2\mathrm{N}}$, with M the number of variable footstep positions within the preview horizon length N,

such that $N \geq M$. Hence, $\mathbf{V}_\mathrm{r}$ and $\mathbf{V}_\mathrm{z}$ must be structured to take into account that $N \geq M$. To do this, the footstep timings are predefined. Knowing this, the rows can be duplicated for the duration of the $j-th$ variable footstep so that:

$$\mathbf{V}_\mathrm{r} = \begin{bmatrix} \mathbf{0} & \dots & \dots & \mathbf{0} \\ \vdots & \dots & \dots & \vdots \\ \mathbf{0} & \dots & \dots & \mathbf{0} \\ \mathbf{R}_{\mathbf{r}_0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \dots & \vdots \\ \mathbf{R}_{\mathbf{r}_0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{R}_{\mathbf{r}_0} & \mathbf{R}_{\mathbf{r}_1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{\mathbf{r}_0} & \mathbf{R}_{\mathbf{r}_1} & \dots & \mathbf{R}_{\mathbf{r}_{\mathrm{M\text{-}1}}} \end{bmatrix}, \quad \mathbf{V}_\mathrm{z} = \begin{bmatrix} \mathbf{R}_{\mathbf{r}_0} & \dots & \mathbf{0} & \dots & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots & \dots & \dots & \vdots \\ \mathbf{0} & \dots & \mathbf{R}_{\mathbf{r}_0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \mathbf{0} & \mathbf{R}_{\mathbf{r}_1} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \dots & \dots & \mathbf{R}_{\mathbf{r}_\mathrm{M}} \end{bmatrix}. \tag{3.17}$$

Note that the foot rotation matrices are separately predetermined/precomputed to preserve linearity [47].

For a walking pattern generator with automatic footstep placement, we can define:

$$\mathbf{x} = [\tilde{\mathbf{u}}^\top \breve{\mathbf{r}}^\top]^\top. \tag{3.18}$$

This vector, which contains the CoM jerk and footstep positions, will be the argument of the optimization problem, formulated below. With $\mathbf{x}$, (3.13) and (3.15) can be rewritten as:

$$\begin{cases} \tilde{\mathbf{c}} = \begin{bmatrix} \mathbf{U}_\mathrm{u} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}} \\ \breve{\mathbf{r}} \end{bmatrix} + \mathbf{U}_\mathrm{c} \hat{\mathbf{c}}_0, \\ \breve{\mathbf{z}} = {\mathbf{V}_\mathrm{z}}^\top \begin{bmatrix} \mathbf{O}_\mathrm{u} & -\mathbf{V}_\mathrm{r} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}} \\ \breve{\mathbf{r}} \end{bmatrix} + {\mathbf{V}_\mathrm{z}}^\top \left(\mathbf{O}_\mathrm{c} \hat{\mathbf{c}}_0 + \mathbf{O}_\mathrm{f} \tilde{\mathbf{f}} - \mathbf{v}_{\mathbf{r}_0} \right). \end{cases} \tag{3.19}$$

More concisely, (3.19) can be written as:

$$\begin{aligned} \tilde{\mathbf{c}} &= \mathbf{S}\mathbf{x} + \mathbf{s}, \\ \breve{\mathbf{z}} &= \mathbf{S}_\mathrm{z}\mathbf{x} + \mathbf{s}_\mathrm{z}. \end{aligned} \tag{3.20}$$

In the following, we use this system to formulate walking as an optimization problem.

3.2.2 Walking as an optimization problem

For now, we reconstruct a WPG similar to the one in [47], which we will refer to as the *standard* walking pattern generator. It can be described as:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & w_1 \left\| \tilde{\mathbf{c}}_\mathrm{v} - \tilde{\mathbf{c}}_{\mathrm{vref}} \right\|^2 + w_2 \left\| \tilde{\mathbf{u}} \right\|^2 + w_3 \left\| \breve{\mathbf{z}} \right\|^2 \\ \text{subject to} \quad & \underline{\breve{\mathbf{z}}} \leq \breve{\mathbf{z}} \leq \overline{\breve{\mathbf{z}}} \\ & \underline{\breve{\mathbf{r}}} \leq \breve{\mathbf{r}} \leq \overline{\breve{\mathbf{r}}}, \end{aligned} \tag{3.21}$$

with the dynamics of $\tilde{\mathbf{c}}$ and $\breve{\mathbf{z}}$ evolving according to (3.20). This WPG has three objectives: tracking a reference CoM velocity, minimizing CoM jerk and minimizing the local ZMP position. There are two constraints: upper and lower bounds for the local ZMP and footstep positions. We can use standard quadratic programming (QP) solvers to obtain the optimal value of $\mathbf{x}$ for this problem. For clarity and brevity, the original variables are shown, but their explicit formulation as functions of $\mathbf{x}$ are given below, along with the details of each objective and constraint of the optimization problem.

The first control objective allows the robot to track a given reference velocity. This provides a simple interface for controlling the walking behavior. To do this, the CoM velocity components need to be extracted from the full CoM state. This can be done using a selection matrix:

$$\tilde{\mathbf{c}}_{\mathrm{v}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{c}} = \mathbf{S}_{\mathrm{v}}\mathbf{x} + \mathbf{s}_{\mathrm{v}}. \tag{3.22}$$

The control objective can then be written as:

$$\left\| \tilde{\mathbf{c}}_{\mathrm{v}} - \tilde{\mathbf{c}}_{\mathrm{vref}} \right\|^2 = \left\| \mathbf{S}_{\mathrm{v}}\mathbf{x} + \mathbf{s}_{\mathrm{v}} - \tilde{\mathbf{c}}_{\mathrm{vref}} \right\|^2, \tag{3.23}$$

and $\tilde{\mathbf{c}}_{\mathrm{vref}}$ is designed by the user or provided by some higher-level algorithm (e.g., a path planner).

The second control objective is to minimize the CoM jerk. Although this is not strictly necessary, it was shown to improve performance in [47]. The aim is to smoothen the control input. Since it is part of the argument, all that is needed is to use a selection matrix such that:

$$\left\| \tilde{\mathbf{u}} \right\|^2 = \left\| \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x} \right\|^2. \tag{3.24}$$

The third and last objective consists in minimizing the local ZMP position. It can be interpreted as minimization of the distance between the ZMP positions and the feet centers. Since the ZMP from Eq.(3.20) is defined with respect to the feet centers, this amounts to giving a vector of zeros as a reference, so that the objective function is:

$$\left\| \breve{\mathbf{z}} - \mathbf{0} \right\|^2 = \left\| \breve{\mathbf{z}} \right\|^2 = \left\| \mathbf{S}_{\mathrm{z}}\mathbf{x} + \mathbf{s}_{\mathrm{z}} \right\|^2. \tag{3.25}$$

The idea behind this objective is that we prefer a change in the foot landing position over a change in the ZMP. This gives a better *stability margin*, since unknown disturbances could push the ZMP away from the target. Since one of the control constraints is to keep the ZMP within the support polygon, the most robust ZMP target location under unknown disturbances is in the middle of these bounds.

The first constraint ensures that the ZMP remains within the support polygon (with some security margins), i.e., within some lower and upper bounds (denoted by the lower and upper bar):

$$\underline{\breve{\mathbf{z}}} \leq \breve{\mathbf{z}} \leq \overline{\breve{\mathbf{z}}}. \tag{3.26}$$

Using the second equation in (3.20) to expose the argument we have:

$$\underline{\breve{\mathbf{z}}} - \mathbf{s}_z \leq \mathbf{S}_z \mathbf{x} \leq \overline{\breve{\mathbf{z}}} - \mathbf{s}_z. \tag{3.27}$$

Finally, the second constraint introduces simple bounds for the feet positions:

$$\underline{\breve{\mathbf{r}}} \leq \breve{\mathbf{r}} \leq \overline{\breve{\mathbf{r}}}. \tag{3.28}$$

As it is part of the argument, we only need to select it as:

$$\underline{\breve{\mathbf{r}}} \leq \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} \leq \overline{\breve{\mathbf{r}}}. \tag{3.29}$$

This constraint is optional when considering the walking pattern generator by itself. However, it plays an important role when considering the whole body control problem. It is a simple way to represent whole body constraints in the WPG.

Another relevant aspect in Eq.(3.21) concerns the weights, that are used to combine the objective functions. In cases where the objectives compete with each other, these weights slightly alter the expected performance. That is: a higher w_1 allows better reference velocity tracking, a higher w_2 reduces motion of the CoM and a higher w_3 allows less ZMP movement. The default weights used in this work are: $w_1 = 0.5, w_2 = 0.05, w_3 = 25$.

Looking at the WPG formulation of Eq.(3.21), it is not immediately apparent how the external wrench affects the final performance of the walking pattern generator. Returning to Eq.(3.13), note that the external wrench only appears in the ZMP formulation. It then affects the WPG in two areas: the ZMP centering objective, and the ZMP constraint. The first interpretation is that the external wrench changes the ZMP value, as explained in [51]. However, we can also say that the external wrench reshapes the ZMP bounds as :

$$\begin{aligned} \underline{\breve{\mathbf{z}}} &\leq \breve{\mathbf{z}}' + \breve{\mathbf{z}}_f \leq \overline{\breve{\mathbf{z}}} \\ \underline{\breve{\mathbf{z}}} - \breve{\mathbf{z}}_f &\leq \breve{\mathbf{z}}' \leq \overline{\breve{\mathbf{z}}} - \breve{\mathbf{z}}_f \end{aligned} \tag{3.30}$$

where $\breve{\mathbf{z}}_f$ represents the ZMP offset resulting from the external wrench and $\breve{\mathbf{z}}'$ the *classic* ZMP (in the absence of external wrench). This is inline with the aforementioned *quadruped model* representing a humanoid and human carrying an object together. The effect of the external wrench on the ZMP centering objective, is to further change the foot landing position based on this ZMP change.

The final missing piece to complete the WPG is the generation of *swing foot* trajectories. This will be the subject of the next subsection.

3.2.3 Swing foot trajectories

Note that the value of $\breve{\mathbf{r}}$ resulting from (3.21) gives us the future foot landing position only. Therefore, we need to generate a trajectory from the last foot position to this new position. In walking, since there is no flight phase (both feet never leave

the ground together), we can define the foot that is off the ground as the *swing foot* without ambiguity. We choose to construct its trajectory based on a cubic polynomial. The 3DOF (x, y, z) are decoupled and defined independently, but follow a similar form:

$$r_{\text{sw}} = a_3 t^3 + a_2 t^2 + a_1 t + a_0, \tag{3.31}$$

where t is the time, r_{sw} is the component of the swing foot position (x, y, or z) and a are the coefficients to find. The first derivative of this polynomial is:

$$\dot{r}_{\text{sw}} = 3a_3 t^2 + 2a_2 t + a_1. \tag{3.32}$$

We can use the initial conditions at $t_0 = 0$ to obtain:

$$\begin{aligned} a_0 =& r_{\text{sw0}}, \\ a_1 =& \dot{r}_{\text{sw0}}. \end{aligned} \tag{3.33}$$

We can also use the final conditions at $t = t_{\text{swf}}$ to obtain:

$$\begin{aligned} r_{\text{swf}} =& a_3 t_{\text{swf}}^3 + a_2 t_{\text{swf}}^2 + a_1 t_{\text{swf}} + a_0, \\ \dot{r}_{\text{swf}} =& 3a_3 t_{\text{swf}}^2 + 2a_2 t_{\text{swf}} + a_1. \end{aligned} \tag{3.34}$$

Then, combining Eq.(3.34, 3.33) we can find the remaining coefficients:

$$\begin{aligned} a_2 =& -\frac{2\dot{r}_{\text{sw0}} t_{\text{swf}} + 3r_{\text{sw0}} + \dot{r}_{\text{swf}} t_{\text{swf}} - 3r_{\text{swf}}}{t_{\text{swf}}^2}, \\ a_3 =& \frac{\dot{r}_{\text{sw0}} t_{\text{swf}} + 2r_{\text{sw0}} + \dot{r}_{\text{swf}} t_{\text{swf}} - 2r_{\text{swf}}}{t_{\text{swf}}^3}. \end{aligned} \tag{3.35}$$

Therefore, to describe this trajectory, we need to define: $r_{\text{sw0}}, \dot{r}_{\text{sw0}}, r_{\text{swf}}, \dot{r}_{\text{swf}}, t_{\text{swf}}$. The total time for the swing foot trajectory t_{swf} can be predefined and is by definition equal to the time for a single support stance (only one foot in contact with the ground). Typically, we set $t_{\text{swf}} = 0.7sec$. We also define the trajectory to start and end in a resting state: $\dot{r}_{\text{sw0}} = 0, \dot{r}_{\text{swf}} = 0$. Furthermore, $r_{\text{sw0}}, r_{\text{swf}}$ are known and generated from the WPG, within $\check{\mathbf{r}}$. Lastly, we need to generate a trajectory for the z DOF. To this end, we divide the z DOF into two trajectories: one going up, the other going down. This can be parametrized simply by a stepping height r_{sth}. For the first half of the total time: $r^z_{\text{sw0}} = 0, r^z_{\text{swf}} = r_{\text{sth}}$. For the second half: $r^z_{\text{sw0}} = r_{\text{sth}}, r^z_{\text{swf}} = 0$. In the default case, we use stepping height $r_{\text{sth}} = 0.07$ meters.

In summary, the $\check{\mathbf{r}}$ resulting from (3.21) is fed to (3.35) and (3.33), to define the swing foot trajectory, according to (3.31), and complete walking pattern generation. Fig. 3.1 shows an example of the swing foot moving along the generated trajectory.

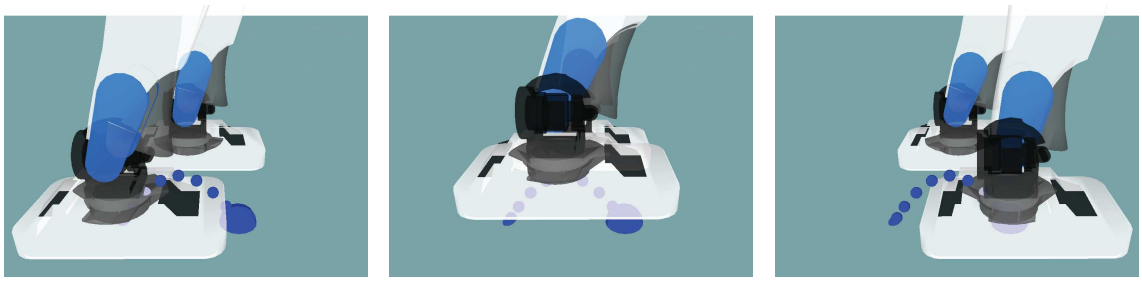


Figure 3.1: A visualization of the parameterized swing foot trajectory

3.3 Specific use cases in physical collaboration

Chapter 2 focused on physical collaboration in the context of manipulation. In this section, the same principles are used directly in walking pattern generation. The formulation of (3.21) serves as the base, from which we can create variations that are better suited to the different types of physical interaction. In particular, we hereby revisit the leader and follower modalities, and show how each one drastically changes the way the external wrench term influences the walking pattern generator.

3.3.1 Walking pattern generator for a follower robot

A follower robot acts based on the leader's intention. This intention can be represented by the external wrench that the leader applies to the robot. Hence, a follower WPG must generate motions that are function of the external wrench applied by the leader (generally, human). Previous works [49, 50] have used a simple damping control, providing the reference velocity as:

$$\dot{c}_{\text{vref}} = \frac{f}{b}, \tag{3.36}$$

within control objective (3.23). We can extend this to perform slightly more complex following behaviors by using a full mass-spring-damper model:

$$f = m\ddot{c} + b\dot{c} + kc. \tag{3.37}$$

Based on (3.37), an admittance control task in the x, y plane can be mapped to the WPG. Since the WPG aims at driving the CoM position, we will not consider the torque of $\tilde{\mathbf{f}}$, but only f^{x} and f^{y}, in the admittance task. Recalling that the state $\tilde{\mathbf{c}}$ contains accelerations, velocities and positions of the CoM, one simply needs to define an appropriate impedance parameter matrix $\mathbf{G}_{\text{mbk}}$. An appropriate selection matrix $\mathbf{S}_{\text{f}}$ is also needed to select only the $f^{\text{x}}, f^{\text{y}}$ components, so that the controller will aim at minimizing:

$$\left\| \mathbf{G}_{\text{mbk}}\tilde{\mathbf{c}} - \mathbf{S}_{\text{f}}\tilde{\mathbf{f}} \right\|^2, \tag{3.38}$$

with:

$$\mathbf{G}_{\mathrm{mbk}} = \begin{bmatrix} k_{\mathrm{x0}} & b_{\mathrm{x0}} & m_{\mathrm{x0}} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{\mathrm{y0}} & b_{\mathrm{y0}} & m_{\mathrm{y0}} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & k_{\mathrm{yN}} & b_{\mathrm{yN}} & m_{\mathrm{yN}} \end{bmatrix},$$
$$\mathbf{S}_{\mathrm{f}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.39}$$

This term will replace the velocity objective used in prior works [49, 50], so the optimization problem can be written as:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & w_1 \left\| \mathbf{G}_{\mathrm{mbk}} \tilde{\mathbf{c}} - \mathbf{S}_{\mathrm{f}} \tilde{\mathbf{f}} \right\|^2 + w_2 \left\| \tilde{\mathbf{u}} \right\|^2 + w_3 \left\| \breve{\mathbf{z}} \right\|^2 \\ \text{subject to} \quad & \underline{\breve{\mathbf{z}}} \leq \breve{\mathbf{z}} \leq \overline{\breve{\mathbf{z}}} \\ & \underline{\breve{\mathbf{r}}} \leq \breve{\mathbf{r}} \leq \overline{\breve{\mathbf{r}}} \end{aligned} \tag{3.40}$$

There are two important points about the external wrench term in the MPC. Firstly, recall from (3.13) that the external wrench terms appearing in the MPC are the future ones $\hat{\mathbf{f}}_1 \dots \hat{\mathbf{f}}_{\mathrm{N}}$. If we use feedback from a force/torque sensor, what is obtained is the current external wrench $\hat{\mathbf{f}}_0$. Therefore, we somehow need to predict its future values. A well-designed prediction model will enable *proactive behaviors* in physical collaboration. The second important aspect is that, since the wrench is expressed in the CoM frame, whereas a sensor produces values in its own frame, a frame transformation is necessary. Here, for the sake of simplicity, we neglect relative motion between sensor and CoM frames, and consider quasi-static motion, when doing such frame transformation.

3.3.2 Walking pattern generator for a leader robot

For leading, a clear and independent intention is necessary. This intention can be represented by a reference behavior that is independent from the follower's actions. We propose two ways for formulating the leader intention in the WPG.

A first idea is to have the leader robot follow a desired trajectory. For instance, in [49], the leader behavior was generated by another human operator, that provided the reference velocity commands $\dot{c}_{\mathrm{vref}}$ needed in (3.23). If the full trajectory is known beforehand, a better objective for the leader may be to track this trajectory directly, rather than to merely track the reference velocity. Such objective can be formulated in the operational space [84]. For the CoM this can be written as:

$$\ddot{c} = \ddot{c}_{\mathrm{ref}} + b(\dot{c}_{\mathrm{ref}} - \dot{c}) + k(c_{\mathrm{ref}} - c), \tag{3.41}$$

with b, k two positive scalar gains. This can be reformulated as an objective function:

$$\left\|(\tilde{\mathbf{c}}_{\text{aref}} - \tilde{\mathbf{c}}_{\text{a}}) + b(\tilde{\mathbf{c}}_{\text{vref}} - \tilde{\mathbf{c}}_{\text{v}}) + k(\tilde{\mathbf{c}}_{\text{pref}} - \tilde{\mathbf{c}}_{\text{p}})\right\|^2. \tag{3.42}$$

To be concise, an appropriate gain matrix can be used, similar in structure to $\mathbf{G}_{\text{mbk}}$ of the follower WPG, so that the objective function can be written as:

$$\left\|\mathbf{G}_{\text{mbk}}(\tilde{\mathbf{c}}_{ref} - \tilde{\mathbf{c}})\right\|^2. \tag{3.43}$$

Another idea for designing the leader robot WPG, is to include the external wrench in the optimization argument, to expand it as: $\mathbf{x} = [\tilde{\mathbf{u}}^\top \ \check{\mathbf{r}}^\top \ \tilde{\mathbf{f}}^\top]^\top$. This changes (3.19) into:

$$\begin{cases} \tilde{\mathbf{c}} = \begin{bmatrix} \mathbf{U}_{\text{u}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}} \\ \check{\mathbf{r}} \\ \tilde{\mathbf{f}} \end{bmatrix} + \mathbf{U}_{\text{c}}\hat{\mathbf{c}}_0, \\ \check{\mathbf{z}} = {\mathbf{V}_{\text{z}}}^\top \begin{bmatrix} \mathbf{O}_{\text{u}} & -\mathbf{V}_{\text{r}} & \mathbf{O}_{\text{f}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}} \\ \check{\mathbf{r}} \\ \tilde{\mathbf{f}} \end{bmatrix} + {\mathbf{V}_{\text{z}}}^\top (\mathbf{O}_{\text{c}}\hat{\mathbf{c}}_0 - \mathbf{v}_{\mathbf{r}_0}). \end{cases} \tag{3.44}$$

Note that only the x and y components of the wrench are in $\tilde{\mathbf{f}}$, whereas $\mathbf{O}_{\text{f}}$ also contains $f^{\text{z}}_{\text{ext}}$, which will therefore have to be predefined over the preview horizon. Expression (3.44) can be concisely written as Eq.(3.20), so that the derivation of objectives and constraints is analogue to that case. Since here, the force is part of the ZMP expression, this may allow the robot to balance itself by applying the appropriate forces. For safety, the applied wrench may need to be constrained:

$$\underline{\tilde{\mathbf{f}}} \leq \tilde{\mathbf{f}} \leq \overline{\tilde{\mathbf{f}}}, \tag{3.45}$$

and/or minimized as:

$$\left\|\tilde{\mathbf{f}}\right\|^2. \tag{3.46}$$

If both options (CoM trajectory tracking, and minimal external wrench) are chosen, the optimization problem can be written as:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & w_1 \left\|\mathbf{G}_{\text{mbk}}(\tilde{\mathbf{c}}_{\text{ref}} - \tilde{\mathbf{c}})\right\|^2 + w_2 \left\|\tilde{\mathbf{u}}\right\|^2 + w_3 \left\|\check{\mathbf{z}}\right\|^2 + w_4 \left\|\tilde{\mathbf{f}}\right\|^2 \\ \text{subject to} \quad & \underline{\check{\mathbf{z}}} \leq \check{\mathbf{z}} \leq \overline{\check{\mathbf{z}}} \\ & \underline{\check{\mathbf{r}}} \leq \check{\mathbf{r}} \leq \overline{\check{\mathbf{r}}} \\ & \underline{\tilde{\mathbf{f}}} \leq \tilde{\mathbf{f}} \leq \overline{\tilde{\mathbf{f}}} \end{aligned} \tag{3.47}$$

Having the external wrench in the argument implies that this result needs to be tracked by the whole-body controller. To do this, we can use an admittance controller on the hands with force sensors. The same quasi-static assumption for the frame transformation is used here.

3.4 Simulation tests

We tested the three WPG formulations (*standard* (3.21), *follower* (3.40), and *leader* (3.47)) in several simulations, prior to the real robot experiments.

We begin with the *standard* WPG, without any external wrench. This serves as our benchmark, and is illustrated by the image sequence of Fig. 3.2. The behavior is essentially equivalent to that of the WPG in [47].

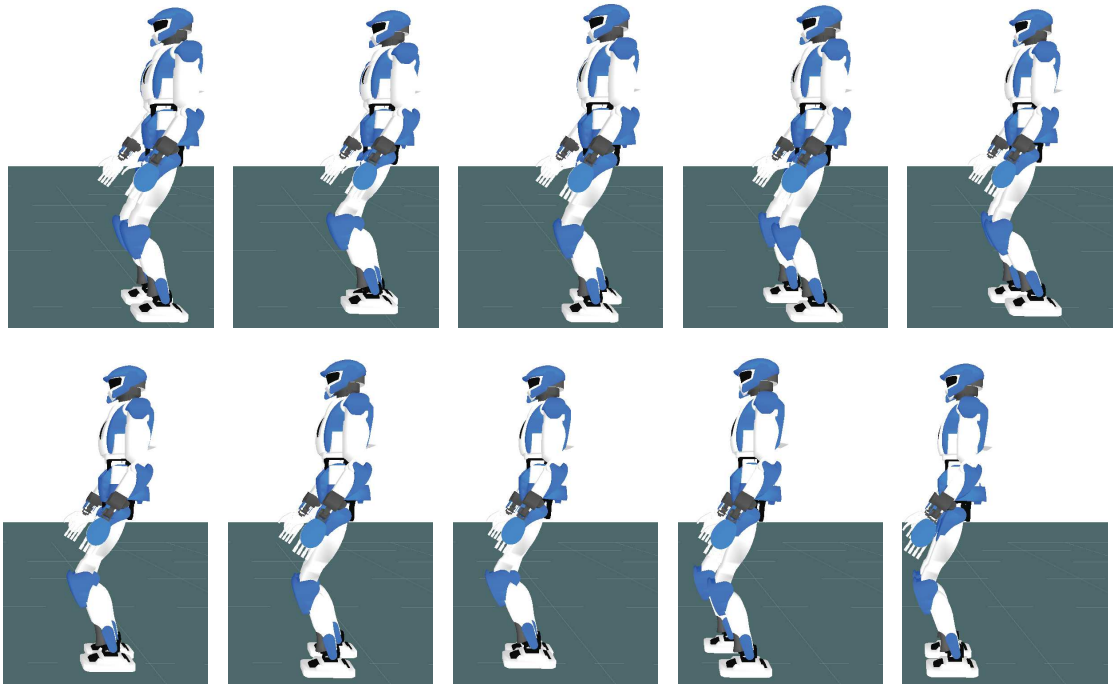
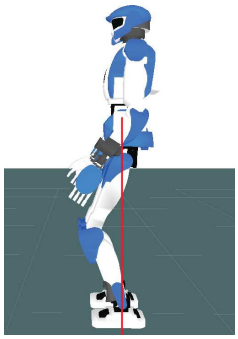


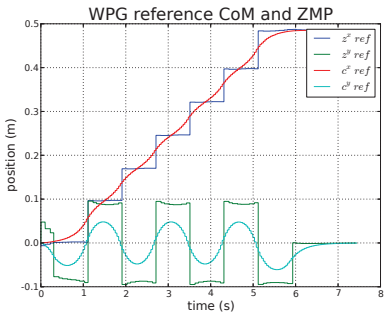
Figure 3.2: Simulations of a *standard* walk with no external wrench, and with desired CoM velocity of 0.1 m/s. Figures run left to right starting from the top, then the bottom.

To highlight the difference when taking into account the external wrench, we add an external constant force of 30 N (towards the left of the figure, i.e., pushing the robot forward) in the simulation. Plots of the generated CoM, ZMP, and footsteps are shown in Fig. 3.3. Along with it, we show the end posture, which is a result of tracking the WPG reference with whole body control that will be detailed later, in chapter 4. Note that the plots labeled 0 N (Fig. 3.3 top) correspond to the image sequence of Fig. 3.2. These show that the base implementation follows well the state of the art implementation of [47]. The simulations with external force are shown in the results labeled with 30N (Fig. 3.3 bottom). Firstly, note that there is a different posture highlighted by the vertical red bar from the middle of the feet (added as a visual aid). We can similarly see this offset in the plots of the CoM and ZMP. Note that the ZMP still follows the footsteps generated, and is close to the middle of the support polygon, following one of the objectives in our optimization problem. As

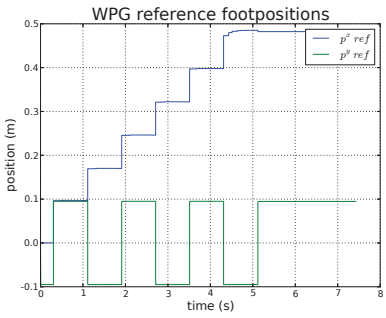
such, it is also well within the defined limits. Finally, notice that a longer footstep was generated at the start, to compensate the initial disturbance. This step reaches the predefined footstep length limit of 0.2 meters. But this is the only time the constraint is active (at the limit): all other cases are well within the predefined limits.



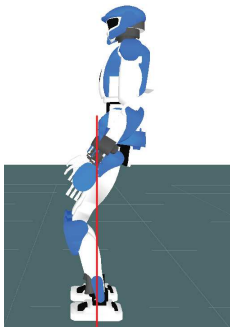
(a) posture, 0N



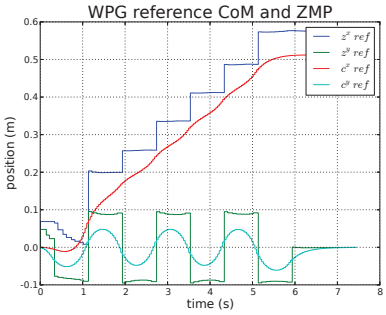
(b) plot of CoM and ZMP, 0N



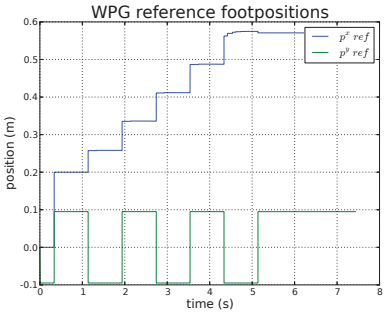
(c) plot of footsteps, 0N



(d) posture, 30N



(e) plot of CoM and ZMP, 30N

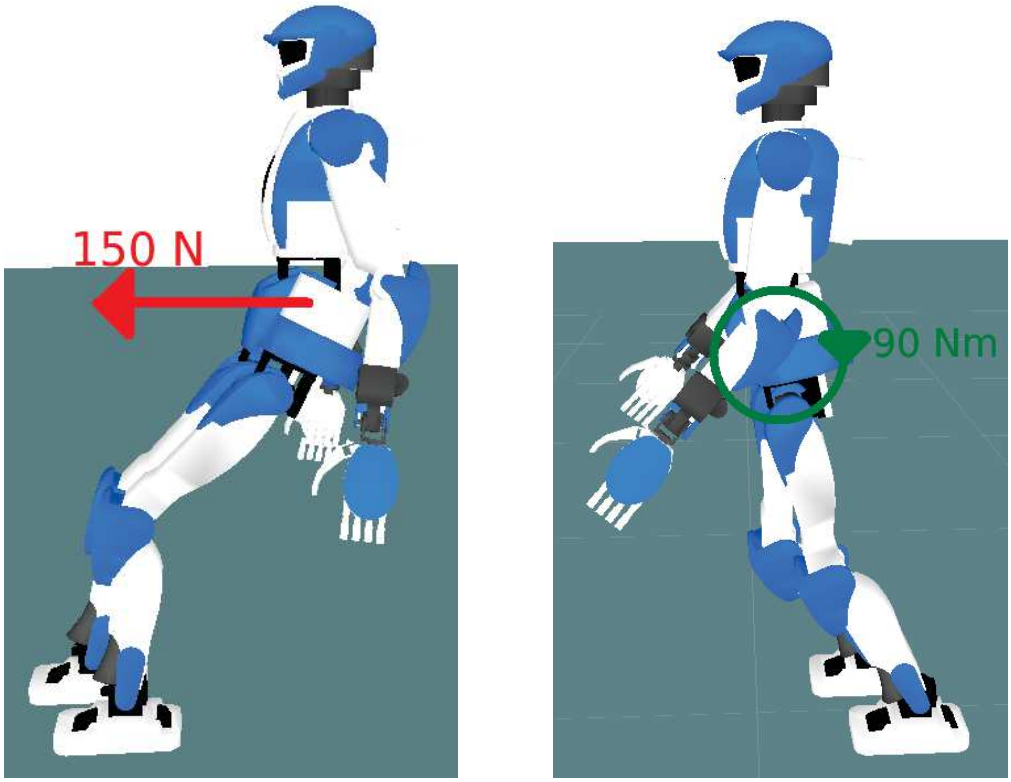


(f) plot of footsteps, 30N

Figure 3.3: Simulations to highlight the reaction to the external wrench of the "standard" WPG. Figures on top (a),(b),(c) show a benchmark without any external wrench (0N), while figures on the bottom (d),(e),(f) show how the WPG reacts to 30 N of external force (going to the left).

Next, we show some more extreme cases, still with the *standard* WPG. Fig. 3.4 highlights the importance of considering the external wrench. The leaning postures in the figure are not preprogrammed, but are the result of tracking the CoM motion and footsteps generated by the WPG, as it compensates the external wrench in the two cases. Note that for all these simulations, the external wrench is also added to the multi-body dynamics to enable the robot to walk in the dynamic simulator.

The next simulation is that of the *follower* WPG of Eq.(3.40). Fig. 3.5 shows the plots of a critically damped follower with a unit mass and a stiffness defined as 50 N/m. This WPG is designed such that the robot does not have its own intention (no desired reference values), but follows the external wrench similarly to the virtual model of a critically damped mass-spring-damper system. That is: it follows the



(a) 150N of forward force, applied on the robot CoM while walking with -0.1m/s reference velocity (backwards)

(b) -90Nm clockwise torque, applied on the robot CoM while walking with 0.1m/s reference velocity (forward)

Figure 3.4: More extreme simulations using the *standard* WPG showing the postural change as a result of taking the external wrench into account

external force when it is applied, but returns to a resting position, defined at the origin of $(0, 0)$, when the external force is released. Several variations of this virtual model can be implemented. For example, a pure damping model (with null mass and stiffness) was used in [10, 50, 85, 86]. Another example is an *oscillating* system with only mass and stiffness, and null damping. Furthermore, more complex virtual models can be defined in a similar manner.

The last simulations are for the *leader* WPG of Eq.(3.47). For the first test, we show that we are able to track a given reference trajectory. This is shown in Fig. 3.6, where the trajectory is drawn in blue. It was defined using a B-spline with points: (0,0), (2,0.5), (1,1) and a duration of 60 seconds. The B-spline is chosen here for simplicity, although our formulation allows tracking of any reference, as long as the position, velocity and acceleration are available. The trajectory is given in 2D (the heading is constant here) and may allow us to leverage some trajectory generation literature from the mobile robotics field. Notice that it allows us to abstract issues such as the swaying of the CoM, and explicit footstep planning. However, care must be taken in designing the trajectory since the WPG constraints

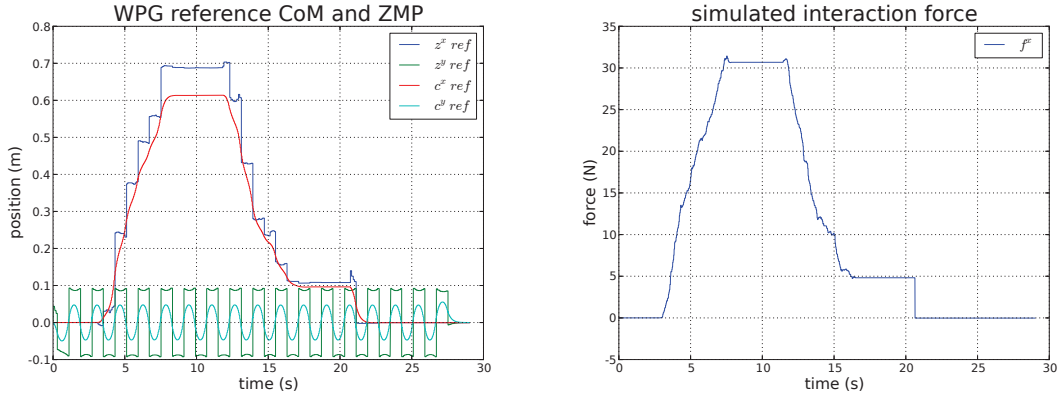


Figure 3.5: Simulation data for a critically damped follower with stiffness = 50 N/m and mass = 1 kg.

are just approximations of the whole-body constraints. We may be able to benefit from better integration of these, as proposed in [87].

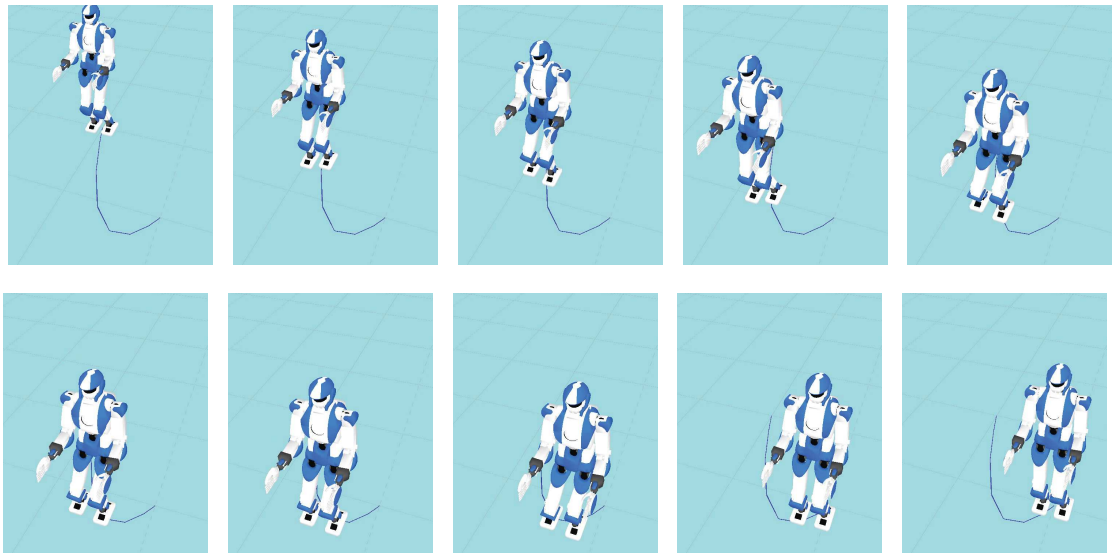


Figure 3.6: Simulations of a *leader* walking sequence following a B-spline trajectory shown by the blue curve.

For the next test of the *leader* formulation, we want to emphasize the usage of the external wrench, for the humanoid balance. In this test, we increase, in the optimization problem, the weight that penalizes the CoM motion. This implies that the robot will use the external wrench, instead of 'swaying' its CoM, to maintain its balance. Furthermore, forces and torques are constrained to be ± 20 N and Nm, respectively. Lastly, the leader trajectory is a B-spline driving the robot 1m forward, in 10 seconds. Figure 3.7 shows the results. Firstly, the plot in x shows that the

trajectory was respected by arriving at 1 meter in 10 seconds. Next, the plot in y shows that the CoM oscillation in the y axis is minimized greatly. Disregarding the start and end phase, the peaks are around 0.01 meters whereas the benchmark of Fig. 3.3 has peaks around 0.05 meters. In place of this, the generated reference external wrench, shown in the last plot, needs to be applied. We anticipate that in future works this property will be very useful in the case of collaborative carrying, as the CoM sway of the robot can disturb the human partner.

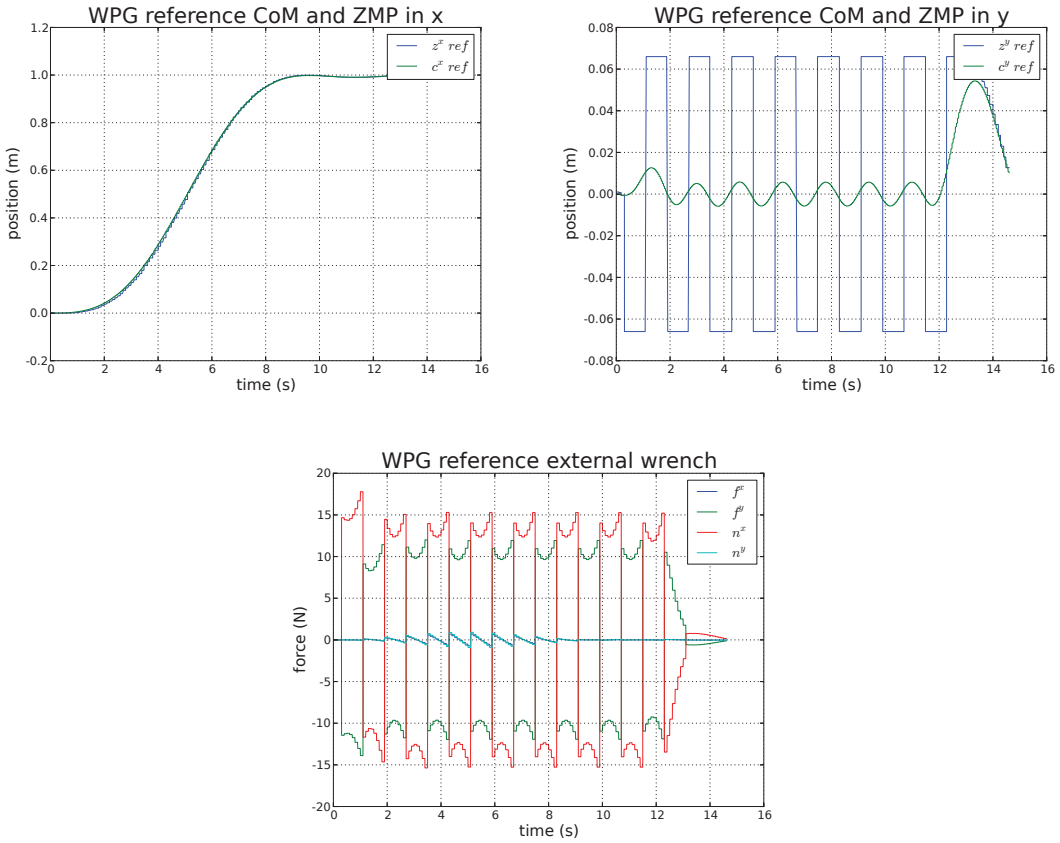


Figure 3.7: Simulation data for a *leader* with a trajectory of 1m forward in 10 seconds.

3.5 Real-robot experiment and results

After verification on simulations, we also test the three WPGs on the real-robot. We propose a switching scenario similar to the one in [49], to show the two follower and leader WPG formulations, respectively (3.40) and (3.47), on the real robot. The robot starts out as a follower, and due to some condition, it switches to being a leader. Some plausible switching conditions are:

- interaction force limits are exceeded,
- workspace limits are exceeded,
- balance is lost (this can be determined, for instance, through capturability limits [45])

In [49], the switch was made at a predetermined time. Here, a maximum interaction force (25 N) is the switching condition, although several conditions (as the ones outlined above) can be used together. The interaction force is applied by a human operator pulling/pushing the robot right hand. For simplicity, the follower was implemented with damping only. The leader trajectory consists in going back 0.2 m in 10 s; although this trajectory is short, it is sufficient to validate the WPG. The torques are constrained to be 0 Nm, whereas forces are allowed to be in the range $\pm 20N$, to simplify the force control problem for one hand.

Figure 3.8 shows a photo during this test, while Fig. 3.9 shows the references, generated by the WPG, for the CoM, ZMP and foot positions, along with the interaction forces. The switching condition occurs around the 50 second mark. After this, the leader trajectory of going back 0.2 m in 10 s is executed. The results show that, even in the presence of real sensor noise, the robot can follow the intent of the human leader. During the leader phase, note that the generated reference force is quite low (gain tuning here was more coarse than in the simulations). This allows the robot to balance itself, without relying on force control. This choice arose from the difficulty in implementing a high-fidelity force control on a position controlled robot.

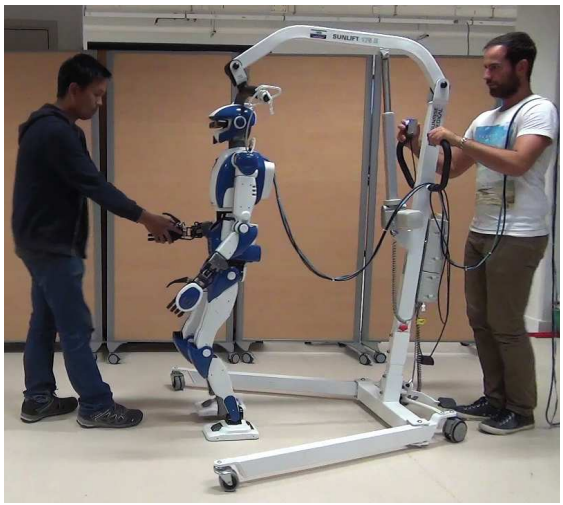


Figure 3.8: Screenshot from the video of the real-robot experiment

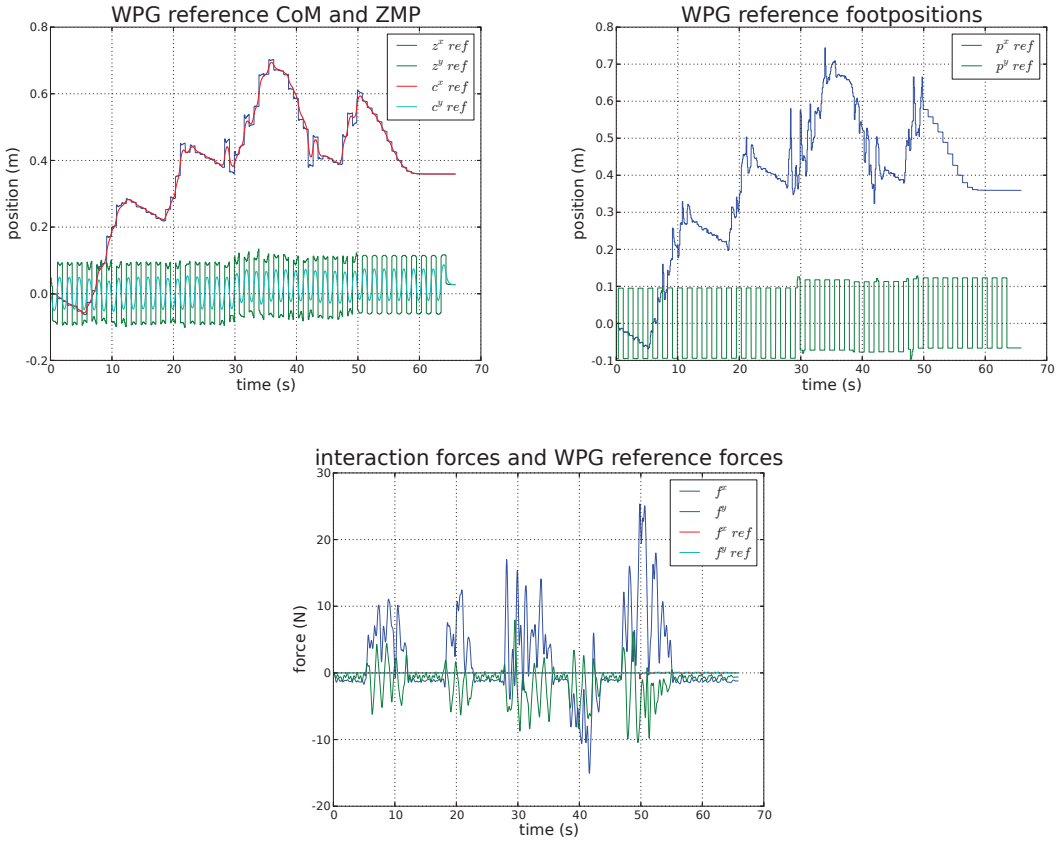


Figure 3.9: Data from the leader-follower switching experiment on the real robot.

Chapter 4

Whole-body control for collaborative carrying

In the previous sections, the focus was on very specific parts of the collaborative carrying task: arm motions utilizing vision and haptic data in chapter 2, Center of Mass and foot motions for walking in chapter 3. These need to be executed in a coordinated manner on a humanoid robot. Therefore, a whole-body controller which can gather all the active tasks and produce corresponding feasible joint motions is needed.

This chapter describes how we worked towards whole-body control in general, with a particular focus on the collaborative carrying task. The chapter starts by observing how human teams collaboratively carry objects, eventually building a taxonomy (Sect. 4.1). The aim is to take inferences and inspiration, that will be used for building the whole-body controller of a humanoid robot. Furthermore, the taxonomy of collaborative carrying can define several cases with only a few descriptors. This allows us to focus on a few cases, knowing that methods can be generalized to all of the other cases under the taxonomy. However, even these few cases are still complex. These need to be broken down into simpler parts for ease of formulating the whole-body control problem. A state machine (described in Sect. 4.2) is used to do this, so that each state can be formulated as an optimization problem. Finally, in Section 4.3, we describe the optimization framework in detail.

4.1 A taxonomy of collaborative carrying

Carrying objects in collaboration with a human partner in various postures and situations is a problem that is rich, unexplored and has a high potential for practical application. Several situations can be envisioned in building sites, rescue sites, disaster sites, etc. We will first focus on various carrying schemes and taxonomies between pairs of humans. From this, we want to gain insights for programming a humanoid. The end goal is to program a humanoid to effectively take the role of one of the human partners in such tasks. Figure 4.1 shows several real examples

of human collaborative carrying, with the corresponding simulations using human avatars.

Figure 4.1: Collaborative carrying examples with human avatars mimicking the corresponding postures

The figure shows the broadness and applicability of this skill. Furthermore, it serves as a useful visualization for breaking down the task. Although the examples may seem very diverse, any collaborative carrying scenario simply boils down to two important components:

1. object being carried,

2. agents (humans and/or humanoid robots) carrying the object.

From these descriptors, we can already form a taxonomy. However, it would not be useful because the object and agents are generally determined a priori. That is, we consider the problem of **having already a team of agents, whose goal is to move a specified object to another location**. We assume that neither the object nor the agent composition can be changed afterwards. For this problem, a components-based taxonomy is useless. However, we can consider the relationship between the components:

1. object-agent relation,

2. agent-agent relation.

The *object-agent relation* is simply defined by how the agents grasp the object. The *grasp type* used by each agent is a useful taxonomy. We can observe that the same object can be grasped in different ways, for example the cylinder in Figure 4.1. There will also be cases where the same grasp type is used for different objects. The *object-agent relation* is elaborated later, in subsection 4.1.2.

The *agent-agent relation* is not as straightforward as the object-agent relation. Several possible relationships fit this category, and can be useful to define a taxonomy:

- team composition (all humans, all robots, ratio in mixed groups, etc.),
- team/agent skill in task execution,
- team coordination (leader, follower, equal collaborators),
- communication modalities used (gesture/vision, speech/audition, haptics),
- agents' relative pose.

Firstly, team composition is not relevant in our problem statement (as explained above). The remaining four relationships are all viable descriptors for creating a taxonomy. In this work, we prefer a descriptor that allows a quick and easy classification of unknown cases. For this, *agents' relative pose* has been chosen among the four, and will be explained in the subsection that follows.

4.1.1 Agents' relative pose

The agents' relative pose can be used to classify any given collaborative carrying scenario. The relative pose itself, has two components, that can be used independently: translation and rotation. These are quantitative descriptors, that can precisely describe each collaborative carrying case. However, using these quantities directly is too specific, thus inappropriate for classifying the various scenarios. Instead, we can use a range of values for classification. The main issue is in finding meaningful range boundaries. For this, we can consider another descriptor of the agent-agent relation, i.e., the communication modalities. In particular, we relate haptic communication to relative translation, and vision to relative rotation, as will be explained hereby.

The translation/distance between the agents can simply be classified as near or far. A meaningful range boundary can be defined by considering if direct haptic communication (touch) is possible or not, so that:

- near: other agent is physically reachable,
- far: other agent is out of reach.

For the rotation, we can define meaningful boundaries by considering vision, more precisely the Field Of View (FOV). To describe the rotational range, let us first define the *nominal* FOV as the FOV when the agent is in a neutral resting position. The *extended* FOV is defined as the FOV of the agent as it looks around, by moving its body to some extent (i.e., without changing stance). With this, we can classify agents as facing:

- front: other agent is in the nominal FOV,
- side: other agent is not in the nominal FOV, but within the extended FOV,
- back: other agent is not in the extended FOV.

These rotation descriptors should be applied to each agent relative to the other. Considering a team with only two agents, the permutations can be tabulated as shown in Table 4.1. This table shows the possible relative rotations of two agents

Agent rotation	Front	Side	Back
Front	Common	Rare	Common
Side	Rare	Common	Rare
Back	Common	Rare	Extremely Rare

Table 4.1: Table of agents' relative rotation for 2 agents in a carrying task

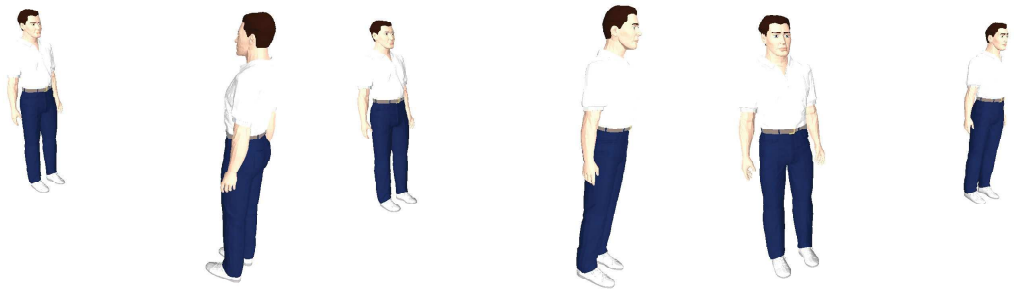
along with their frequency in *real-world* cases. Note that the table is symmetric with respect to the diagonal (e.g., front-side is equivalent to side-front). Thus, there are six classes in the taxonomy that considers only the agents' relative rotations. These classes are illustrated in Fig. 4.2 with human avatars.

4.1.2 Grasp types

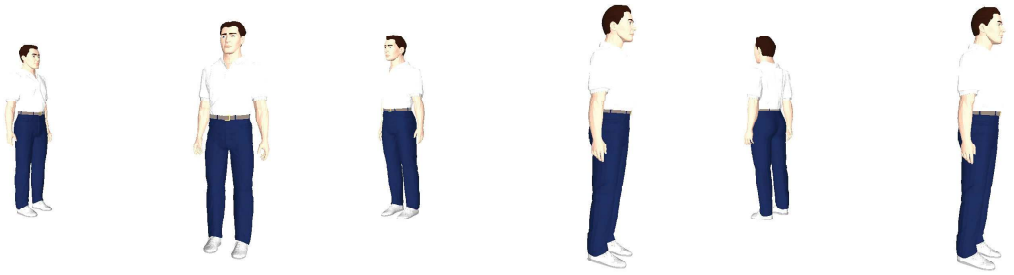
Independently from the *agent-agent relation*, a taxonomy of grasp types is considered which characterizes the *object-agent relation*. For this classification, we define two broad grasp types: *hand grasps* and *body grasps*.

Hand grasps are those with contact points located uniquely on the hand/gripper. These are the ones used to carry the table, stretcher and bucket in Fig. 4.1. These grasps are the subject of most of the existing robotics literature on grasping. A detailed taxonomy of hand grasps and its analysis are presented in [88]. In particular, [88] notes that there are three main aspects to be considered when choosing a grasp:

1. task to do with the object after grasping,
2. object to be grasped,
3. gripper to be used.



(a) front-front (b) front-back/back-front (c) side-side



(d) front-side/side-front (e) side-back/back-side (f) back-back

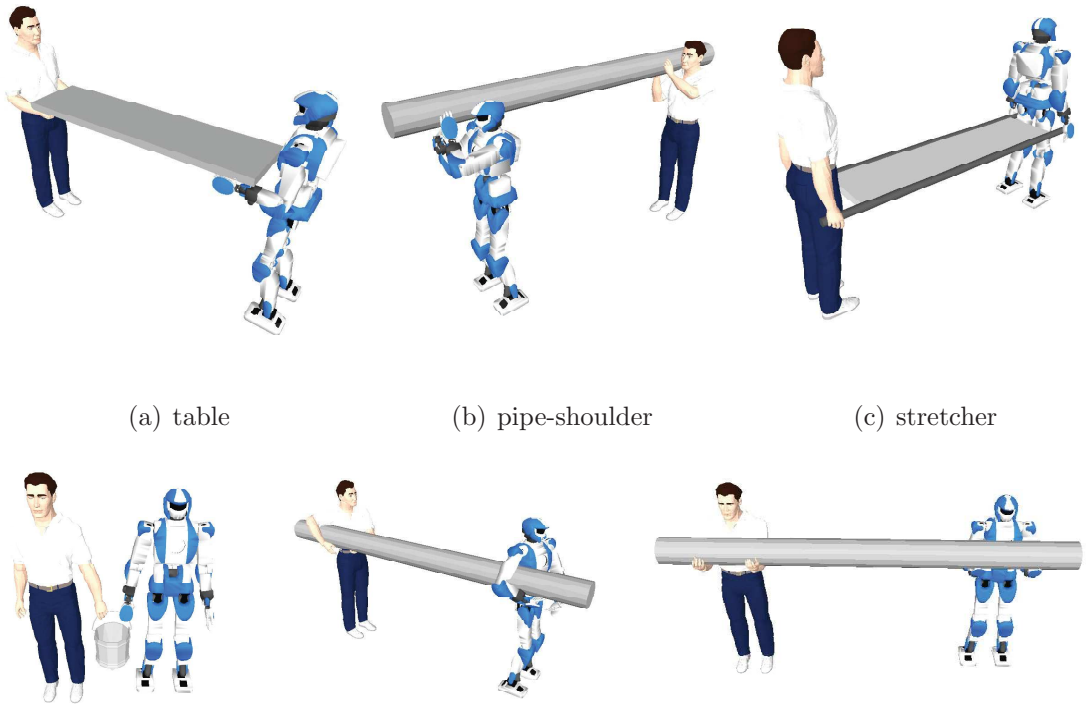
Figure 4.2: Simulation of a human dyad in the taxonomy of agents' relative rotations

The first two points are well-defined in our problem statement. The third is particularly interesting. In fact, we can consider another class of *grippers* which characterizes the second classification: *body grasps*. Indeed, we define body grasps as those that utilize contact points on body parts not limited to the hand. For example, the arms, shoulder and torso can be considered. In Fig. 4.1, this is the case when carrying the steel pipe or wooden logs. We observe that the object to be grasped requires the use of this kind of grasp. Generally, to grasp large objects, or objects without *natural handles*, the whole body can be used to form a larger *gripper*. The weight and shape of these object can determine the design of the body grasp postures.

Finally, there are two important observations for this taxonomy. Firstly, even though body grasps are more general, hand grasps are preferred if possible. Similarly in robotics, *grasping* is closely related to hand design [88]. We can infer that for humans, the hand is our preferred *tool* for grasping and we have learned to use it with a lot of skill. This observation is useful for programming a humanoid robot to emulate humans. Secondly, it is common for both agents to use the same grasp type, as shown in Fig. 4.1. Although there is no restriction for such, we can infer that if both agents have the same capabilities, and if the object is symmetric, then using the same grasp type is the logical conclusion for both agents.

4.1.3 Applying the taxonomy to human-humanoid teams

To recapitulate on this section, we first show how to apply the taxonomy to some simulations of human-humanoid teams, seeking to emulate Fig. 4.1. After this, we briefly discuss the usage of this taxonomy in the next sections. The simulations are shown in Fig. 4.3.



(a) table (b) pipe-shoulder (c) stretcher

(d) bucket (e) pipe-side (f) pipe-front

Figure 4.3: Different carrying scenarios with the HRP4 and a human avatar

All the scenarios of Fig. 4.3 can be classified according to the proposed taxonomy. Each case can be specified by classifying it according to the three criteria, i.e.: relative translation, relative rotation and grasp type. Specifically, for the six scenarios of Fig. 4.3:

(a) table: far, front-front, hand grasp

(b) pipe-shoulder: far, front-back, body grasp

(c) stretcher: far, front-back, hand grasp

(d) bucket: near, side-side, hand grasp

(e) pipe-side: far, front-back, body grasp

(f) pipe-front: far, side-side, body grasp

This shows the ease of classifying given cases. But more importantly, we are concerned with the practical implications of the taxonomy towards programming a humanoid robot. The direct relation is that the relative pose and grasp type describe the goal of the humanoid when going towards the object and when grasping it. However, the relative pose has a deeper meaning. As already mentioned in subsection 4.1.1, its classes are related to sensing, and to the availability of communication modalities (vision and haptic). Therefore, the required pose of the robot with respect to the human will largely affect what type of sensors are needed, and the best way to use them. For instance, consider a humanoid robot equipped with tactile sensing on the arms and/or shoulders. In a *near relative translation* scenario, these can be used for communication (e.g., tapping the arm as a signal to start/stop walking). On the contrary, *far relative translation* scenarios immediately disallow this possibility. For vision, a *front-facing* robot has the possibility to read gestures or even body language of the human, yielding better following behaviors. Instead, a *back-facing* robot has the possibility to be more aware of the surroundings and of the environment (e.g., it can use visual navigation algorithms), yielding better leading behaviors. Although this may imply that each class in the taxonomy must be programmed differently, the base actions of all classes remain the same. The cited examples for communication modalities present possible enhancements rather than the defining characteristics of collaborative carrying. The next section aims at formulating the generic collaborative carrying algorithm. Along the way, we also show the role of the taxonomy in making claims of generalization.

4.2 Decomposing a task into a Finite State Machine

A task is an abstraction without any upper bounds on its complexity. This makes it difficult to program a robot for any general task. To handle this, we must decompose complex tasks into subtasks that will be easier to program. Throughout this thesis, we tackle the task of collaborative carrying, and this section aims at decomposing it into proper subtasks.

Outside the context of programming robots, we humans often describe complicated tasks by using simpler subtasks. Take, for example, user manuals, step-by-step instruction guides, or even cooking recipes. A similar decomposition process must be done, until each step is simple enough, to be used for programming a robot. Formally, we can use a Finite State Machine (FSM) to describe the whole task, with the subtasks as states. A useful decomposition is one where the states can be easily written as optimization problems, which we use to control the robot body (whole body control). A guideline for doing this is to first consider the state transitions. These must allow a smooth change from one optimization problem to the next. An easy way to do this, is to identify brief periods where the motion is minimal. During these periods, the robot is said to be in a *quasi-static* state. More formally, the

dynamic effects are small enough such that disregarding it has no bad consequence. Another important signifier of state transitions are discrete changes in the contact state of the robot. Considering these, a first decomposition of the collaborative carrying task into an FSM is shown by Fig. 4.4.

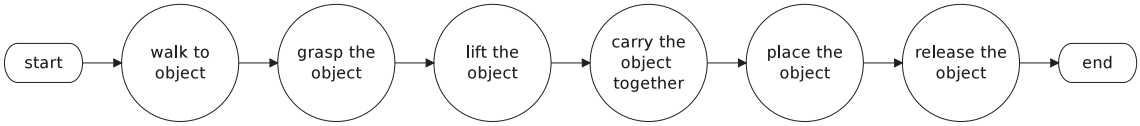


Figure 4.4: FSM for collaborative carrying

Notice that there is a natural progression between states, with transitions where the robot is quasi-static. Furthermore, the conditions for transitioning between states can be automated by checking simple thresholds, for example the distance between the robot and object, or the forces applied to the object when grasping. The transition conditions can be obtained from sensor information, while the thresholds are dependent on several factors (the robot workspace in a given stance, for example).

This decomposition is useful for pointing out how the taxonomy is used. The taxonomy on the agents' relative pose allows us to set a target stance, for the robot to reach, when walking towards the object. It also dictates how to carry the object together. Furthermore, it can serve as a guideline on how to best use the sensors for the duration of the task. The taxonomy on grasp types defines how the object will be grasped, and also how it is held while carrying it together. Although this FSM now has simpler subtasks, it is still not easy to translate some subtasks into optimization problems, for whole body control. We can further develop and decompose two of these, in particular *walking* and *grasping*. Doing this will also indirectly decompose *carrying the object together*. Let us start with walking.

A *dynamic walking motion* does not have any quasi-static phase to be taken advantage of. However, the contact transitions of the feet occur in a predictable pattern. This pattern can be used to define three distinct walking states:

1. left single support,
2. right single support,
3. double support.

Left single support (LSS) corresponds to the state where the left foot is used to support the robot weight (in contact with the ground), while the right foot is in the air. Right single support (RSS) is the inverse state, with the right foot in contact with the ground. Double support (DS) is the state with both feet in contact with the ground. By definition, walking does not have a *flight phase* where both feet leave the ground simultaneously. This is found in other forms of locomotion, such as running or hopping, which we do not consider here. Lastly, we can also use strict timing to define the state transitions (except for the walk start and end). This can simplify

the synchronization between the whole-body control and the WPG from chapter 3. Following these considerations, the FSM for walking is illustrated in Fig. 4.5.

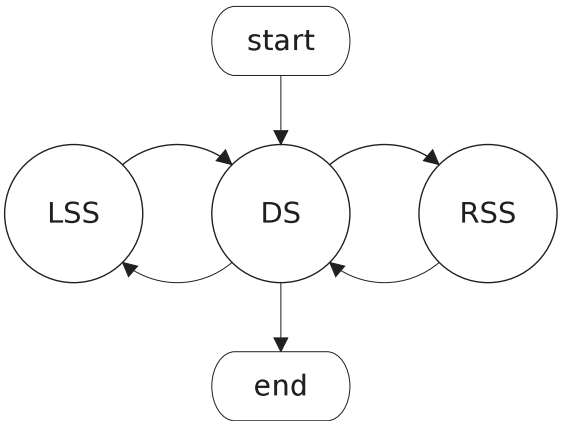


Figure 4.5: FSM for walking

To simplify the decomposition of grasping, we can define a *pregrasp* posture. This describes a state that will serve as a waypoint between grasping the object and the other states of the FSM (see Fig. 4.6).

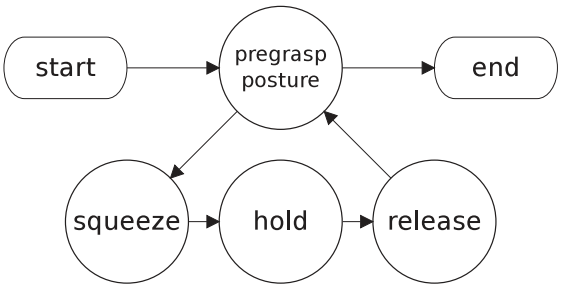


Figure 4.6: FSM for grasping

The definition of the pregrasp posture is tightly linked with grasp synthesis, and can be formalized as a posture generation problem, considering the object and robot properties. It can also be generated by considering caging [89] as a waypoint to grasping the object. Due to added complexity, we do not include this in the scope of this work. Instead of trying to synthesize a pregrasp posture automatically, we can rely on the taxonomy classes, and on observations from humans, to choose particular grasp types and postures for specific objects. This choice can be made beforehand. For example, we chose to design the body grasps shown in Fig. 4.7 to emulate the *pipe-shoulder* and *pipe-front* examples of Fig. 4.3. In these 2 examples, we parameterized the grasp by defining three contact points on the robot and servoing these accordingly. The details of which are explained later on in Section 4.3.

The next state, *squeeze*, simply moves the robot so that the predefined contacts between the robot and object are made. Force or tactile sensors, when available, can

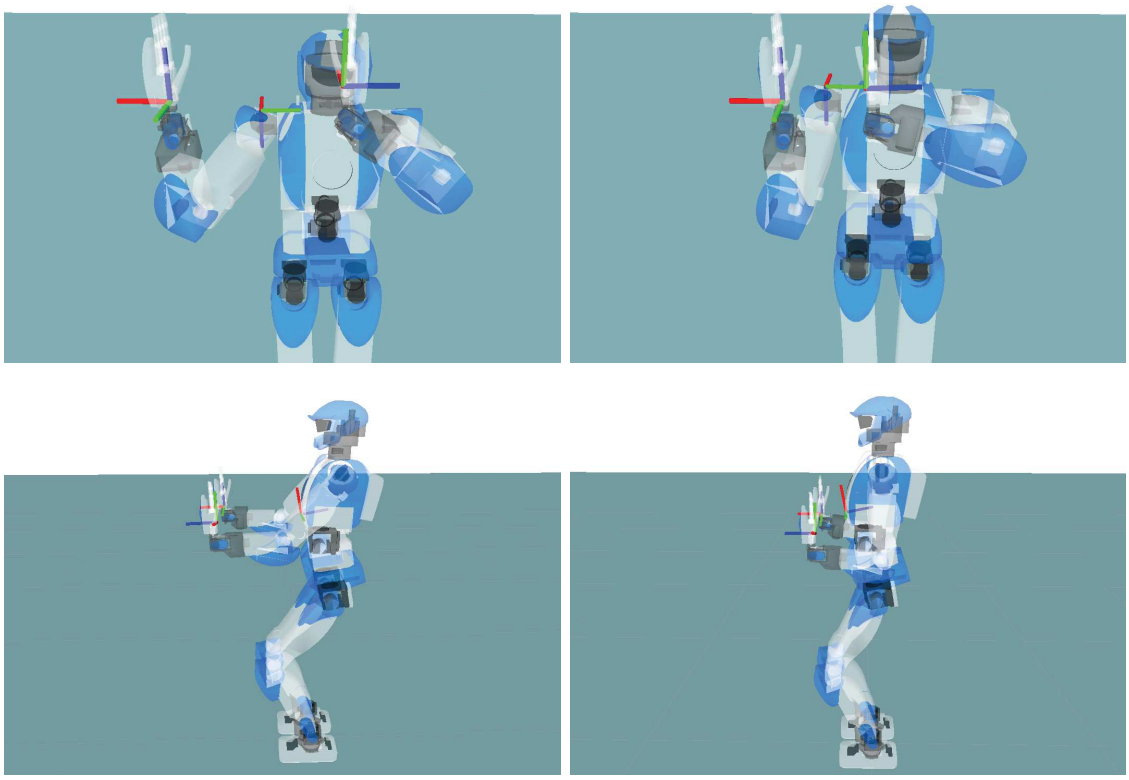


Figure 4.7: Examples of pregrasp postures using "body grasps" (left) and the resulting posture after squeezing (right).

help improve the execution of this state, and better signal the transition to the next state. The *hold* state simply consists in maintaining the contacts between the robot and the object. Finally, the *release* state, is simply the inverse motion of *squeeze*.

By using both the grasping and walking FSMs, we can then create a more detailed version of Fig. 4.4. Each state of this new FSM will be formalized as an individual optimization problem, later in section 4.3.3. This detailed FSM is shown in Fig. 4.8, with the numbers indicating the transition order of a normal execution of the collaborative carrying task. The numbers on the walking transitions were omitted for brevity.

4.3 Whole-body control as an optimization problem

Whole-body control consists in the synthesis of a robot's low-level motor commands (usually in *joint-space*) from the high-level task goals. These high-level goals are defined in the corresponding *operational spaces*. The most common of these is the 3D Cartesian space, another example is the image space. The second problem to

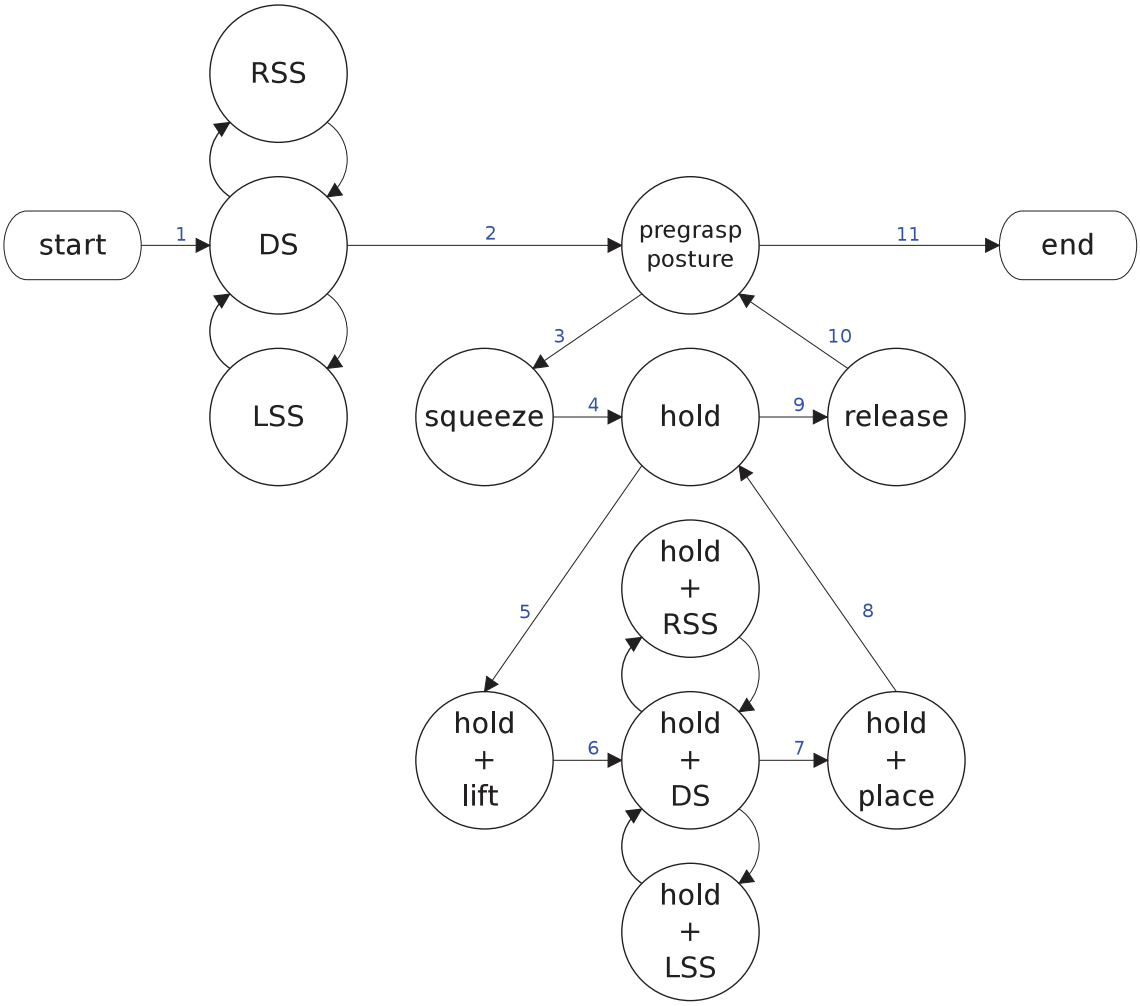


Figure 4.8: Detailed FSM for collaborative carrying

be addressed in whole-body control is handling multiple constraints with varying levels of importance. In this thesis, the whole-body control problem is cast into an optimization problem with these two issues in mind. The optimization-based approach has shown to be effective for whole-body control [33–36]. In contrast with these works, this thesis concentrates on collaborative carrying, and on how each state of the previously designed FSM can be written as an optimization problem. In general, all the cited works seek to find the robot control input, represented by the optimization argument $\mathbf{x}$, such that:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{argmin}} && \text{collection of task functions,} \\ &\text{subject to} && \text{robot constraints} \\ & && \text{environment constraints} \\ & && \text{task constraints.} \end{aligned} \tag{4.1}$$

The different instances of Eq.(4.1) arise from the choices in the implementation of the various components. We hereby discuss such choices.

Firstly, the choice of $\mathbf{x}$ represents how the low level motor commands will be obtained. It also implies what task functions will be viable, since all functions will need to be a function of such argument: $f(\mathbf{x})$. A formulation considering only kinematic tasks and constraints can conveniently choose $\mathbf{x} = \dot{\mathbf{q}}$, while a representation that needs to take into account the dynamics, is better represented by $\mathbf{x} = \ddot{\mathbf{q}}$. Meanwhile, a torque controlled robot may benefit from the choice of $\mathbf{x} = \boldsymbol{\tau}$. Furthermore, it is possible to concatenate these with other variables such as the contact forces (a form which we will use later on).

Second, the choice on how to represent the collection of task functions will determine how the problem of task prioritization will be handled. A simple way to combine tasks is by adding weights such that the total objective function is simply a weighted sum of the different objectives:

$$f_{\text{total}}(\mathbf{x}) = \sum_{i=0}^{n} w_{\text{i}} f_{\text{i}}(\mathbf{x}). \tag{4.2}$$

This is used in [34, 35]. Another method, is to represent the collection by a strict hierarchy, as in [33, 36]. This can be done with several different methods (reviewed in [36]), some of which require to restructure the problem into several optimization problems, to be solved in sequence or by some type of null-space projection operator. In this thesis, both formulations were utilized at different points: the hierarchical formulation of the *Stack of Tasks* [33] for earlier work, and a weighted QP similar to the one used in [35], for later work. This section focuses on the later works, although there is little to no change, if the collaborative carrying tasks are to be used in a hierarchical framework. In fact, although most works distinguish between the two methods, both approaches are not mutually exclusive, and weighted formulations can exist within the hierarchy levels. Furthermore, up to a certain degree, it is possible to emulate strict hierarchies in a weighted optimization. One way to do this is by choosing the weights so that:

$$w_{\text{i}} \min(f_{\text{i}}(\mathbf{x})) > w_{\text{i-1}} \max(f_{\text{i-1}}(\mathbf{x})),$$

where task i has a higher priority than task $i-1$, and min() and max() represent the expected/acceptable minimal and maximal error values, respectively. For example, $\min(f(\mathbf{x}))$ can represent the minimum tolerance or convergence threshold while $\max(f(\mathbf{x}))$ can be a bound on the maximum possible task error.

Lastly, the numeric optimization algorithm to solve the problem must be properly chosen. A good overview of standard methods is presented in [90]. This choice strongly affects what task functions are viable, and how they should be designed. However, we need to consider the trade-offs. For example, non-linear solvers are more general but tend to be much slower. A popular choice is quadratic programming (QP) [34–36], which allows to use the Euclidean or L^2 norm for defining the task functions.

4.3.1 The base Quadratic Programming formulation

The *base* formulation of the optimization problem used here, is largely based on [35] and by extension on [34]. We make the same design choices, by considering a weighted quadratic programming problem. The argument is chosen to be:

$$\mathbf{x} = \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix}. \tag{4.3}$$

In this section, $\mathbf{q}$ includes the definition of the free-flyer as in Eq.(3.1),(3.2). The other part of the argument, $\boldsymbol{\lambda}$, is the vector of linearized friction cone base weights such that the contact forces can be described as:

$$\mathbf{f}_{\text{contact}} = \mathbf{K}_{\text{fc}}\boldsymbol{\lambda}, \tag{4.4}$$

with $\mathbf{K}_{\text{fc}} \in \mathbb{R}^{3n \times nm}$ a matrix of generators for linearizing the friction cone (n is the number of contact points, and m the number of generators chosen for the linearization).

Then, we can rewrite Eq.(4.1) more specifically as:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & \sum_{i=1}^{n} w_{\text{i}} f_{\text{i}}(\mathbf{x}) + w_{\lambda} \left\|\boldsymbol{\lambda}\right\|^2 \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0} \\ & \underline{\boldsymbol{\tau}} \leq \boldsymbol{\tau} \leq \overline{\boldsymbol{\tau}} \\ & \underline{\mathbf{q}} \leq \mathbf{q} \leq \overline{\mathbf{q}} \\ & \underline{\dot{\mathbf{q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{q}}} \\ & \text{other constraints} \end{aligned} \tag{4.5}$$

We will refer to this as the *base* formulation since it only specifies the most essential and generic parts of the optimization problem which are always used. Notice that most of the constraints in the base formulation are robot limitations, which are assumed known. The details of Eq.(4.5), starting from the objective functions, will be hereby discussed.

For a QP, all the $f_{\text{i}}(\mathbf{x})$ must follow the form:

$$f_{\text{i}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\top}\mathbf{Q}_{\text{qp}}\mathbf{x} + \mathbf{c}_{\text{qp}}^{\top}\mathbf{x}. \tag{4.6}$$

However, it is well-known that objectives expressed in linear form:

$$\mathbf{A}_{\text{ls}}\mathbf{x} = \mathbf{b}_{\text{ls}} \tag{4.7}$$

can be incorporated in a QP, by using the objective function:

$$f_{\text{i}}(\mathbf{x}) = \frac{1}{2}\left\|\mathbf{A}_{\text{ls}}\mathbf{x} - \mathbf{b}_{\text{ls}}\right\|^2 = \frac{1}{2}(\mathbf{A}_{\text{ls}}\mathbf{x} - \mathbf{b}_{\text{ls}})^{\top}(\mathbf{A}_{\text{ls}}\mathbf{x} - \mathbf{b}_{\text{ls}}), \tag{4.8}$$

such that:

$$\mathbf{Q}_{\text{qp}} = \mathbf{A}_{\text{ls}}^{\top}\mathbf{A}_{\text{ls}}, \quad \mathbf{c}_{\text{qp}} = -\mathbf{A}_{\text{ls}}^{\top}\mathbf{b}_{\text{ls}}. \tag{4.9}$$

This implies that, writing the linear form of an objective, will ensure its feasibility in QP form, while the constant $\frac{1}{2}$ can be ignored or considered among the weights. Going back to the base form, the objective functions $f_{\text{i}}(\mathbf{x})$ correspond to the collaborative carrying subtasks, that will be detailed later.

The objective of minimizing $\|\boldsymbol{\lambda}\|^2$ ensures that matrix $\mathbf{Q}_{\text{qp}}$ is positive definite. More precisely, it works together with the posture task described later on to ensure this. This property is useful for solving the QP [90]. This objective function can then be written as:

$$\left\| \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} \right\|^2. \tag{4.10}$$

Let us now focus on the optimization problem constraints.

The first constraint ensures that the contact forces will be inside the friction cone (no slipping) and can be formulated by:

$$\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x} \geq \mathbf{0}. \tag{4.11}$$

The second constraint places bounds on the actuator torques. The torques can be obtained from the canonical equation of a robot mechanism:

$$\boldsymbol{\tau} = \mathbf{H}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \boldsymbol{\tau}_{\mathbf{g}} - \mathbf{J}^{\top}\mathbf{f}_{\text{contact}}, \tag{4.12}$$

with all variables defined in the List of symbols. A standard constraint form can then be obtained as:

$$\underline{\boldsymbol{\tau}} - \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}_{\mathbf{g}} \leq \begin{bmatrix} \mathbf{H} & -\mathbf{J}^{\top}\mathbf{K}_{\text{fc}} \end{bmatrix} \mathbf{x} \leq \overline{\boldsymbol{\tau}} - \mathbf{C}\dot{\mathbf{q}} - \boldsymbol{\tau}_{\mathbf{g}}. \tag{4.13}$$

The third and fourth constraints respectively bound the joint positions and velocities. A simple implementation consists in numerical integration (other methods are possible), so that, for a discrete time interval k:

$$\begin{aligned} \dot{\mathbf{q}}_{\text{k}+1} =& \dot{\mathbf{q}}_{\text{k}} + \ddot{\mathbf{q}}_{\text{k}}\Delta t, \\ \mathbf{q}_{\text{k}+1} =& \mathbf{q}_{\text{k}} + \dot{\mathbf{q}}_{\text{k}}\Delta t + \frac{1}{2}\ddot{\mathbf{q}}_{\text{k}}\Delta t^2. \end{aligned} \tag{4.14}$$

Using these, we can rewrite the constraints as:

$$\begin{aligned} \underline{\dot{\mathbf{q}}} - \dot{\mathbf{q}} \leq \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}\Delta t \leq \overline{\dot{\mathbf{q}}} - \dot{\mathbf{q}}, \\ \underline{\mathbf{q}} - \mathbf{q} - \dot{\mathbf{q}}\Delta t \leq \frac{1}{2}\begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x}\Delta t^2 \leq \overline{\mathbf{q}} - \mathbf{q} - \dot{\mathbf{q}}\Delta t. \end{aligned} \tag{4.15}$$

A significant improvement can be made if these constraints are formulated as limit avoidances instead of simple bounds. This can be achieved by adding avoidance

functions, that change the constraints when they are close to being reached. Then, the third and fourth constraints in the base formulation become:

$$\begin{aligned} \underline{\mathbf{q}} + f(\mathbf{q}, \underline{\mathbf{q}}) \leq \mathbf{q} \leq \overline{\mathbf{q}} + f(\mathbf{q}, \overline{\mathbf{q}}), \\ \underline{\dot{\mathbf{q}}} + f(\dot{\mathbf{q}}, \underline{\dot{\mathbf{q}}}) \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{q}}} + f(\dot{\mathbf{q}}, \overline{\dot{\mathbf{q}}}), \end{aligned} \tag{4.16}$$

where $f(\mathbf{q}, \underline{\mathbf{q}}), f(\mathbf{q}, \overline{\mathbf{q}}), f(\dot{\mathbf{q}}, \underline{\dot{\mathbf{q}}}), f(\dot{\mathbf{q}}, \overline{\dot{\mathbf{q}}})$ are the mentioned avoidance functions. It is also possible to combine joint position and velocity constraints, with avoidance functions, as in [35]. This completes the description of the base formulation.

4.3.2 Reusable objectives and constraints

Aside from the base formulation, there are several tasks and constraints that are recurrent and can be written in re-usable form. First, let us define a task vector in some operational space, $\mathbf{e}$. To use it, we need a function mapping the task vector into joint space such that:

$$\mathbf{e} = f_{\mathbf{e}}(\mathbf{q}). \tag{4.17}$$

Let us assume that this function is twice differentiable such that we can write:

$$\dot{\mathbf{e}} = \mathbf{J}_{\mathbf{e}}\dot{\mathbf{q}}, \tag{4.18}$$

$$\ddot{\mathbf{e}} = \mathbf{J}_{\mathbf{e}}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{\mathbf{e}}\dot{\mathbf{q}}, \tag{4.19}$$

where $\mathbf{J}_{\mathbf{e}}$ is the task Jacobian. For a task vector defined in Cartesian space, $\mathbf{J}_{\mathbf{e}}$ is the classic robot Jacobian.

From Eq.(4.19), we can define a constraint to suppress motion for a part of the robot in contact. For example, we can simply define the task as the contact point on the robot in Cartesian space and constrain the acceleration to be null $\ddot{\mathbf{e}} = \mathbf{0}$ such that:

$$\mathbf{J}_{\mathbf{e}}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{\mathbf{e}}\dot{\mathbf{q}} = \mathbf{0}, \tag{4.20}$$

and the constraint can be written as:

$$\begin{bmatrix} \mathbf{J}_{\mathbf{e}} & \mathbf{0} \end{bmatrix} \mathbf{x} = -\dot{\mathbf{J}}_{\mathbf{e}}\dot{\mathbf{q}}. \tag{4.21}$$

From this formulation, a selection matrix can also be added to release some DOF [35]. This constraint was used in [34]. However, there are some numerical stability issues with using Eq.(4.20) as it is [91]. This can be compensated for by defining:

$$\mathbf{J}_{\mathbf{e}}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{\mathbf{e}}\dot{\mathbf{q}} = f_{\text{cmp}}(\Delta t), \tag{4.22}$$

where $f_{\text{cmp}}(\Delta t)$ is a function aiming to compensate for the numerical integration errors. As this is not the main issue here, we will disregard it for simplicity, although practical implementation requires the use of $f_{\text{cmp}}(\Delta t)$. The interested reader is referred to [91] for further details. From here on, we refer to Eq.(4.21) as the contact constraint.

Contrary to the contact constraint which prevents motion, we often want to servo a certain part of the robot body. Several design choices, with their corresponding implementations allow doing this. In this thesis, we adapt the tracking task in operational space [84]. This is defined as:

$$\ddot{e} = \ddot{e}_{\text{des}} + b(\dot{e}_{\text{des}} - \dot{e}) + k(e_{\text{des}} - e). \tag{4.23}$$

where the subscript *des* denotes a desired reference and b, k define the gains. The tracking task can be expressed as a cost function:

$$f_{\text{tr}}(\mathbf{x}) = \frac{1}{2} \left\| k(\mathbf{e}_{\text{des}} - \mathbf{e}) + b(\dot{\mathbf{e}}_{\text{des}} - \dot{\mathbf{e}}) + (\ddot{\mathbf{e}}_{\text{des}} - \ddot{\mathbf{e}}) \right\|^2, \tag{4.24}$$

where $\mathbf{e}$ denotes a vector of task variables to be tracked. This formulation is so generic that several tasks can use it, e.g.: any physical point on the robot, the Center of Mass, the robot posture, image pixel coordinates, etc. Furthermore, in the absence of suitable reference velocities and accelerations, one can set these to zero. The result is the set-point objective from [34]:

$$f_{\text{sp}}(\mathbf{x}) = \frac{1}{2} \left\| k(\mathbf{e}_{\text{des}} - \mathbf{e}) - b\dot{\mathbf{e}} - \ddot{\mathbf{e}} \right\|^2. \tag{4.25}$$

Gains k and b can be tuned by considering the task dynamics equivalent to those of a mass-spring-damper system with unit mass. For a critically damped system, only k needs to be tuned, with $b = 2\sqrt{k}$. To transform Eq.(4.24) into a QP of the form in Eq.(4.6), we can use Eq.(4.17)(4.18),(4.19), to obtain:

$$\begin{aligned} \mathbf{Q}_{\text{qp}} &= \begin{bmatrix} \mathbf{J}_{\mathbf{e}}^\top \mathbf{J}_{\mathbf{e}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \\ \mathbf{c}_{\text{qp}} &= \begin{bmatrix} -\mathbf{J}_{\mathbf{e}}^\top \left(k(\mathbf{e}_{\text{des}} - f_{\mathbf{e}}(\mathbf{q})) + b(\dot{\mathbf{e}}_{\text{des}} - \mathbf{J}_{\mathbf{e}}\dot{\mathbf{q}}) + \ddot{\mathbf{e}}_{\text{des}} - \dot{\mathbf{J}}_{\mathbf{e}}\dot{\mathbf{q}} \right) \\ \mathbf{0} \end{bmatrix}. \end{aligned} \tag{4.26}$$

From here on, we refer to this formulation as the tracking task. A particular formulation of this task is obtained when it consists in joint positioning. In that case, we define it as:

$$\mathbf{e} = \mathbf{q}, \quad \mathbf{J}_{\mathbf{e}} = \mathbf{I}, \tag{4.27}$$

and normally we use the set-point form of Eq.(4.25) to get:

$$f_{\text{pos}}(\mathbf{x}) = \frac{1}{2} \left\| k(\mathbf{q}_{\text{des}} - \mathbf{q}) - b\dot{\mathbf{q}} - \ddot{\mathbf{q}} \right\|^2. \tag{4.28}$$

We refer to this as a *posture task*. Along with the objective of minimizing $\|\boldsymbol{\lambda}\|^2$, this task ensures that $\mathbf{Q}_{\text{qp}}$ is positive definite. Furthermore, the physical meaning of the posture task is to have a *default* configuration of each joint. This implies that its corresponding weight w_{q} normally has a relatively low value. This is useful

when there are few other active tasks. The desired posture $\mathbf{q}_{\text{des}}$ is usually given by a resting state of the robot (i.e. standing). In particular, for the HRP robots, this corresponds to the *half-sitting* posture. It is also possible to define the desired postures depending on the task to be realized.

Lastly, it is also possible to add collision avoidance [35]. This is generally used to avoid self collisions between robot body parts, and is not essential to the overall task. Hence, we decided not to detail it here.

4.3.3 Formalizing the FSM states as optimization problems

Now that we have all the ingredients for the QP, we will go through the collaborative carrying FSM states in order. In this section, we only detail objectives and constraints that are specific to the task. Keep in mind that tasks and constraints from the basic form of Eq.(4.5) are always active. We will begin with the states related to walking.

For a Double Support (DS) state, both feet: $\mathbf{r}_{\text{left}}, \mathbf{r}_{\text{right}}$, must keep the contact with the ground. The CoM is servoed with a trajectory: $\mathbf{c}_{\text{des}}, \dot{\mathbf{c}}_{\text{des}}, \ddot{\mathbf{c}}_{\text{des}}$, which is obtained from the WPG of chapter 3. Optionally, we can also add a set-point orientation for the torso $\boldsymbol{\theta}_{\text{tsdes}}$ to have an upright posture while walking. With this, the whole-body optimization problem is then:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & w_{\text{c}} f_{\text{tr}}(\mathbf{x}, \mathbf{c}_{\text{des}}, \dot{\mathbf{c}}_{\text{des}}, \ddot{\mathbf{c}}_{\text{des}}) + w_{\text{ts}} f_{\text{sp}}(\mathbf{x}, \boldsymbol{\theta}_{\text{tsdes}}) + w_{\text{q}} f_{\text{pos}}(\mathbf{x}, \mathbf{q}_{\text{des}}) \\ \text{subject to} \quad & \ddot{\mathbf{r}}_{\text{left}} = \mathbf{0} \\ & \ddot{\mathbf{r}}_{\text{right}} = \mathbf{0}. \end{aligned} \tag{4.29}$$

Similarly, we can formalize the single support states, by defining a support foot $\mathbf{r}_{\text{sup}}$, and a swing foot $\mathbf{r}_{\text{sw}}$, and labeling the state as LSS (respectively, RSS) when the the left (right) foot is in support, and the right (left) is swinging. The formulation is similar to (4.29), except that the swing foot tracking task replaces the second contact constraint. The generation of the swing foot trajectory: $\mathbf{p}_{\text{swdes}}, \dot{\mathbf{p}}_{\text{swdes}}, \ddot{\mathbf{p}}_{\text{swdes}}$ was described in Chapter 3. The optimization problem is:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & w_{\text{c}} f_{\text{tr}}(\mathbf{x}, \mathbf{c}_{\text{des}}, \dot{\mathbf{c}}_{\text{des}}, \ddot{\mathbf{c}}_{\text{des}}) + w_{\text{sw}} f_{\text{tr}}(\mathbf{x}, \mathbf{p}_{\text{swdes}}, \dot{\mathbf{p}}_{\text{swdes}}, \ddot{\mathbf{p}}_{\text{swdes}}) + \\ & w_{\text{ts}} f_{\text{sp}}(\mathbf{x}, \boldsymbol{\theta}_{\text{tsdes}}) + w_{\text{q}} f_{\text{pos}}(\mathbf{x}, \mathbf{q}_{\text{des}}) \\ \text{subject to} \quad & \ddot{\mathbf{r}}_{\text{sup}} = \mathbf{0}. \end{aligned} \tag{4.30}$$

For grasping, it is important to start from the *pregrasp* posture. For this, let us define n operational points on the robot body with $i = 1 \ldots n$. The pose of each is then denoted by: $\mathbf{p}_{\text{gr, i}}$. Each of these will have a corresponding pre-grasp pose: $\mathbf{p}_{\text{grdes, i}}$, such that we can formulate n set-point tasks. It is also possible to generate trajectories for these, as with the swing foot, and then use a tracking task. The set-point formulation is used here for simplicity. It is also possible to include collision avoidance with the object [35], or to generate a collision-free trajectory, but these

are omitted, again for simplicity. Going to a pre-grasp posture can then be written as the optimization problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & \sum_{i=1}^{n} w_{\text{gr,i}} f_{\text{sp,i}}(\mathbf{x}, \mathbf{p}_{\text{grdes, i}}) + w_{\text{c}} f_{\text{sp}}(\mathbf{x}, \mathbf{c}_{\text{des}}) + w_{\text{q}} f_{\text{pos}}(\mathbf{x}, \mathbf{q}_{\text{des}}) \\ \text{subject to} \quad & \ddot{\mathbf{r}}_{\text{left}} = \mathbf{0} \\ & \ddot{\mathbf{r}}_{\text{right}} = \mathbf{0}. \end{aligned} \tag{4.31}$$

The same formulation applies to the *release* state. The *squeeze* state is very similar in formulation, but this time $\mathbf{p}_{\text{grdes, i}}$ is defined by the expected contact points on the object. It is possible to use set-point tasks in this state, as well. However, a better task way is to use the available wrench information, $\mathbf{h}_{\text{gr}}$, to do the grasp. For example, a simple guarded-move to signal stopping or some type of force control can be used. In summary, the grasp optimization problem is:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & \sum_{i=1}^{n} w_{\text{gr,i}} f_{\text{gr,i}}(\mathbf{x}, \mathbf{p}_{\text{grdes, i}}, \mathbf{h}_{\text{gr}}) + w_{\text{c}} f_{\text{sp}}(\mathbf{x}, \mathbf{c}_{\text{des}}) + w_{\text{q}} f_{\text{pos}}(\mathbf{x}, \mathbf{q}_{\text{des}}) \\ \text{subject to} \quad & \ddot{\mathbf{r}}_{\text{left}} = \mathbf{0} \\ & \ddot{\mathbf{r}}_{\text{right}} = \mathbf{0}. \end{aligned} \tag{4.32}$$

After successfully squeezing the object, a grasp is maintained by the *hold* state. Here, we choose to formalize this via relative contact constraints between the grasp points. These constraints replace the set-point tasks, so that the optimization problem can be written:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & w_{\text{c}} f_{\text{sp}}(\mathbf{x}, \mathbf{c}_{\text{des}}) + w_{\text{q}} f_{\text{pos}}(\mathbf{x}, \mathbf{q}_{\text{des}}) \\ \text{subject to} \quad & \ddot{\mathbf{p}}_{\text{gr,1}} - \ddot{\mathbf{p}}_{\text{gr,2}} = \mathbf{0} \\ & \vdots \\ & \ddot{\mathbf{p}}_{\text{gr,n-1}} - \ddot{\mathbf{p}}_{\text{gr,n}} = \mathbf{0} \\ & \ddot{\mathbf{r}}_{\text{left}} = \mathbf{0} \\ & \ddot{\mathbf{r}}_{\text{right}} = \mathbf{0} \end{aligned} \tag{4.33}$$

In principle, it is possible to define all the permutations between the grasp points as constraints. However, because of numerical considerations, this becomes a very complex problem for the solver. Instead, we only define $n-1$ contact constraints. These are chosen so that each one is considered without creating a kinematic loop between them. A simple way to do this is to define all the constraints between i and $i+1$ with $i = 1, \ldots, n-1$. This is represented by Eq.(4.33). Moreover, it can be advantageous here to change the target of the posture task to maintain the form of the grasp, while executing other motions.

Furthermore, the hold state is to be realized simultaneously with other states. For the walking states (DS, LSS, RSS), while holding, there are only few changes.

Firstly, the CoM uses a tracking task as in walking states. Secondly, the foot contact constraints are dictated by the walking states, while the grasp contact constraints are always maintained. Thirdly, a swing foot tracking task is added when appropriate. Lastly, the torso orientation task is optionally added.

The last states that need to be described are *lift* and *place*. Once the object is held, it can be considered as part of the robot. We can then define an operational point related to the object, *obj* and servo its pose. Here, we choose to servo it with a set-point task but as before, the motion can be improved with a trajectory generator, possibly including collision avoidance. The optimization formulation is then:

$$\begin{aligned} \underset{\mathbf{x}}{\text{argmin}} \quad & w_{\text{obj}} f_{\text{sp}}(\mathbf{x}, \mathbf{p}_{\text{objdes}}) + w_{\text{c}} f_{\text{sp}}(\mathbf{x}, \mathbf{c}_{\text{des}}) + w_{\text{q}} f_{\text{pos}}(\mathbf{x}, \mathbf{q}_{\text{des}}) \\ \text{subject to} \quad & \ddot{\mathbf{p}}_{\text{gr,1}} - \ddot{\mathbf{p}}_{\text{gr,2}} = \mathbf{0} \\ & \vdots \\ & \ddot{\mathbf{p}}_{\text{gr,n-1}} - \ddot{\mathbf{p}}_{\text{gr,n}} = \mathbf{0} \\ & \ddot{\mathbf{r}}_{\text{left}} = \mathbf{0} \\ & \ddot{\mathbf{r}}_{\text{right}} = \mathbf{0}. \end{aligned} \tag{4.34}$$

4.4 Simulations

The objectives and constraints described in various cases were tested in a series of simulations. Furthermore, these are also heavily used in [91] and in other works of our research group. Thus, in this section, we concentrate on the tasks specific to collaborative carrying.

To begin with, we start with walking. For this, the WPG of chapter 3 is integrated into our whole-body controller. This was partially shown in the results of that chapter. To better illustrate this, the results of the tracking task for the CoM are shown in Fig. 4.9. Recall that the WPG generates a reference CoM position, velocity and acceleration at each instant. Furthermore, since we are running the WPG and whole-body controller at different loop rates (100ms and 5ms respectively), we use a simple Euler integration to interpolate the reference. The plots show that the CoM is tracked well enough. Along with the CoM, the ZMP of the whole-body, and that of the WPG, can be compared. Note that there is currently no explicit task, within the whole-body controller, to track the reference ZMP. However, on the real HRP-4 platform, this is used within a low-level *stabilizer*, which acts as a final closed-loop controller, to ensure balance. This is a common practice in walking robots [37]. Therefore, for the ZMP, we see much more difference between the curves, although our tests (both dynamic simulation, and real tests) show that this does not affect the balance of the robot. Also note that in terms of integration within the whole-body controller, the various WPGs presented in chapter 3 function similarly. The only difference is in needing force control in the leader case, and force sensing in the follower case.

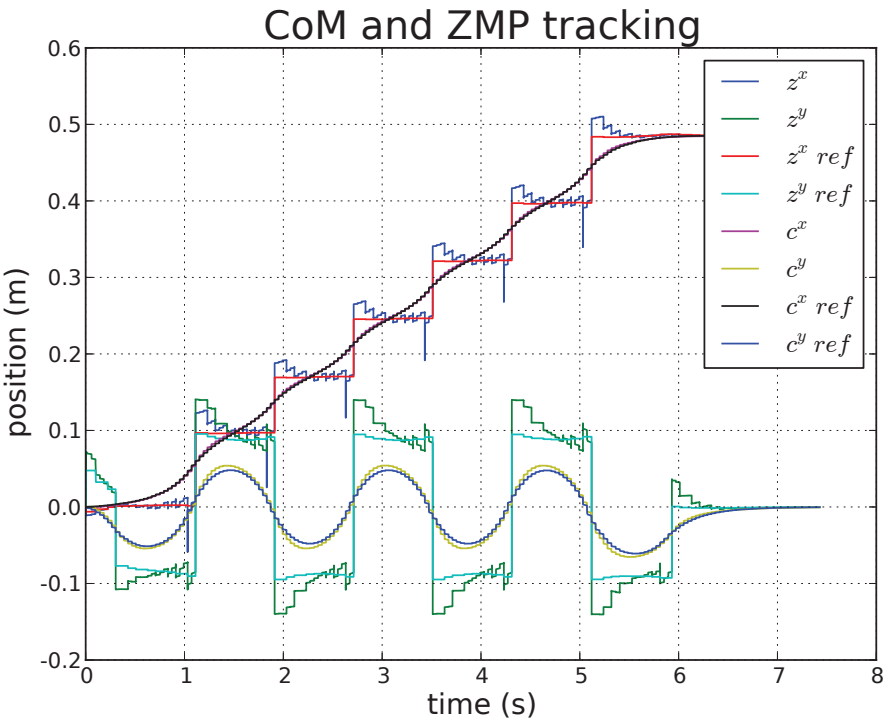


Figure 4.9: Tracking task of the CoM using the WPG-generated reference CoM along with a comparison of the resulting ZMP.

Secondly, we show the performance of the swing foot tracking task while walking. This is depicted in Fig. 4.10, where the trail of the feet (green curve) is shown to follow the generated reference (blue dots). Along with this, the reference frames of the left and right feet are shown. Furthermore, the same trail showing the results of the tracking task also shows the contact constraint on the support foot, since there is no movement of the frame for the foot defined as being in support (the green curve only shows movement of the swing foot).

Next, for the grasping postures, some simulations have been shown in Fig. 4.7. However, because of some unavoidable hardware issues (broken wrist joint) we also had to create *one-handed* versions of these. These are shown in Fig. 4.11, along with a grasping motion of the hand. Note that for the hand of the HRP-4 robot, the thumb has 1 DOF with 1 Motor, and the four other fingers are actuated all together by another motor. Hence, the 4 fingers open and close together during squeezing and this motion is controlled by a single *joint position*. Another point of interest might be the left arm in the front-wrap closing (shown in the middle right figure). This motion is caused by the task on the CoM (to keep the projection on the ground near the center of the feet in this case). Since the squeeze motion moves the chest frame forward, the optimization tries to use the left arm to compensate the CoM task.

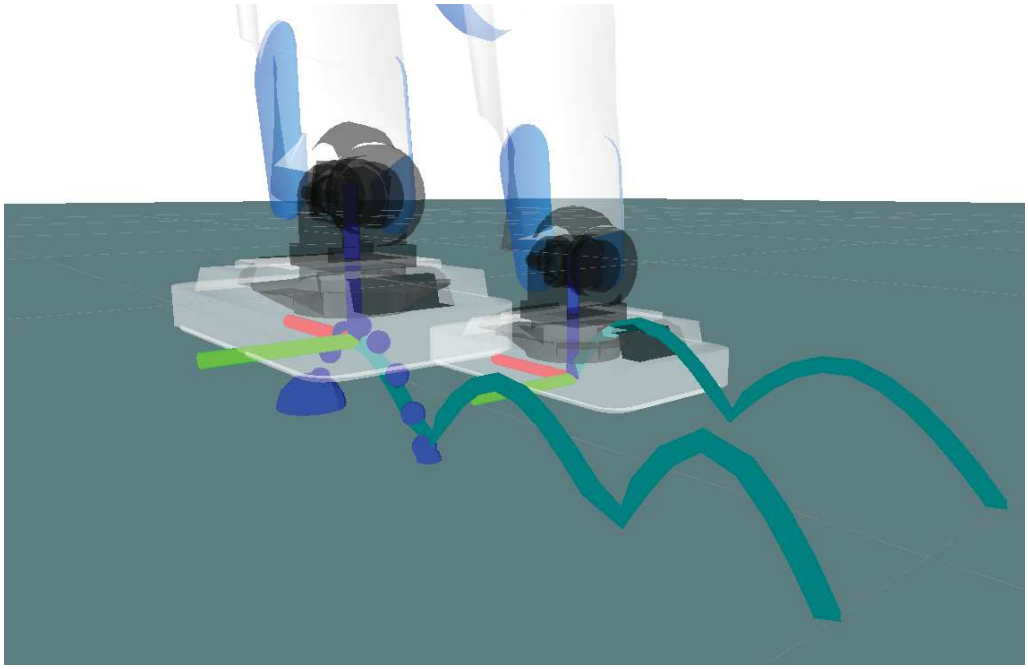


Figure 4.10: Results of the tracking task on the swing foot: left and right foot reference frames, currently generated swing foot reference trajectory (blue dots), past trail of the feet axes (green curve).

Lastly, we show some simulations of walking while holding the body grasps. Figure 4.12 shows an image sequence of walking while holding the front-wrap body grasp while Fig. 4.13 shows an image sequence of walking while holding the shoulder-mount body grasp. These simulations show the full two-hand versions of the grasp. Clearly, any of the WPG may be used together with any of the grasps. The figure only illustrates some chosen examples.

4.5 Real robot experiments and results

After these simulations, we validated some situations on the HRP-4 robot. Due to time constraints, we were not able to do many of the envisioned scenarios from Fig. 4.3. Some preliminary tests were done, as shown in Fig. 4.14. As mentioned before, these use the *one-hand* versions of the body grasps, because of a broken wrist joint. Although we were able to do some preliminary tests, the real experiments introduce some issues which were not present in simulation. Firstly, we only have local force sensing in the wrists of the robot. The other contact points, i.e., the shoulder and chest, had no force sensors. Thus, we had to take this into account when using the sensed data (for example, in the *follower* WPG). Furthermore, there were some minor issues with the grasp stability. However, the force sensing is a deeper issue, and there are several ways to improve the current situation. For instance,

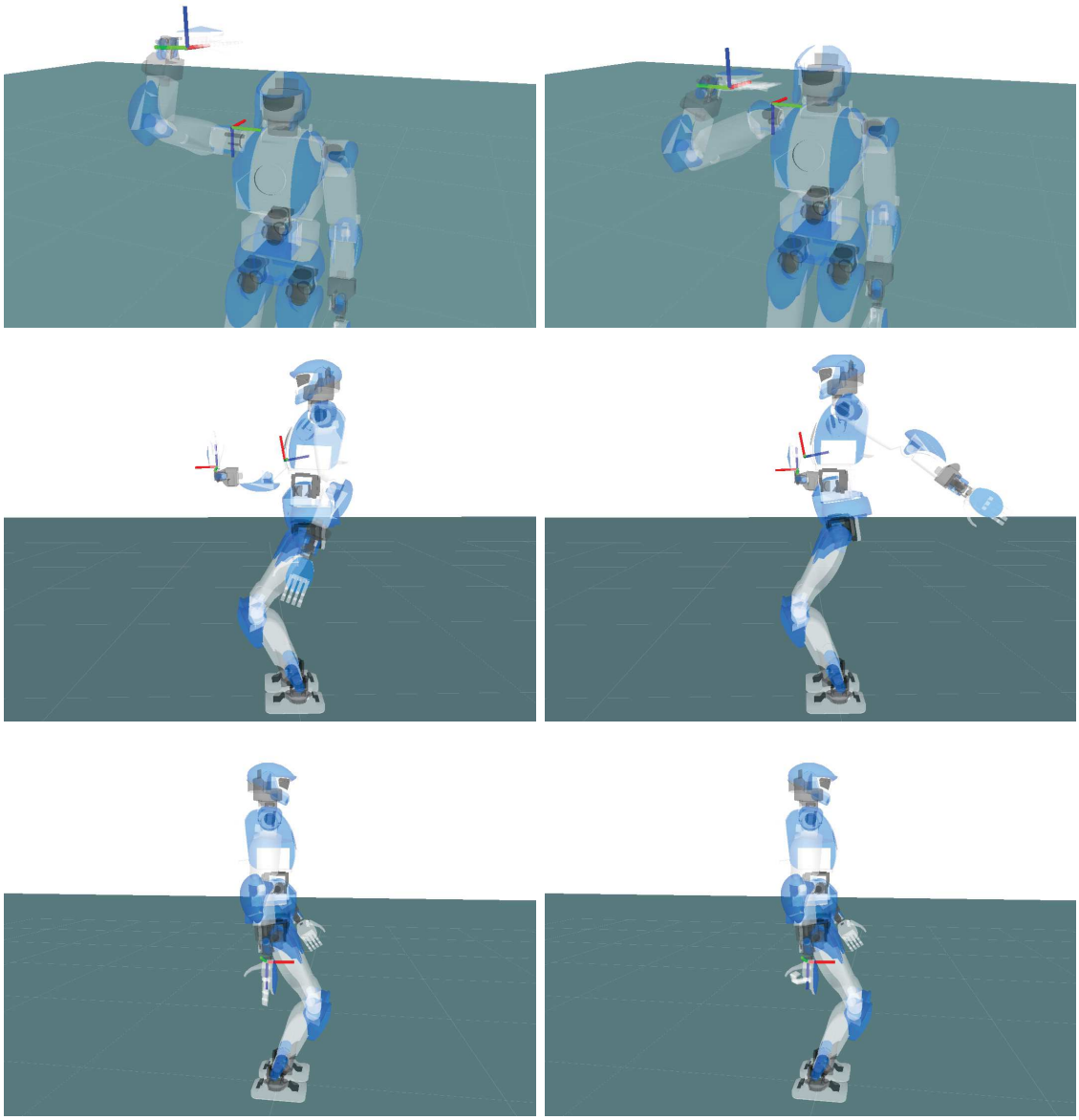


Figure 4.11: Grasping postures (left) along with the posture after squeezing (right). From top to bottom: Shoulder-mounted body grasp (one-handed version), front-wrap body grasp (one-handed version), and right hand grasp.

adding a *skin* over the envisioned contact points, to have some force data, or taking into account other sensed data, e.g., from the IMU, or even from vision, in some specific cases. The other issues are minor, and can be handled given more time.

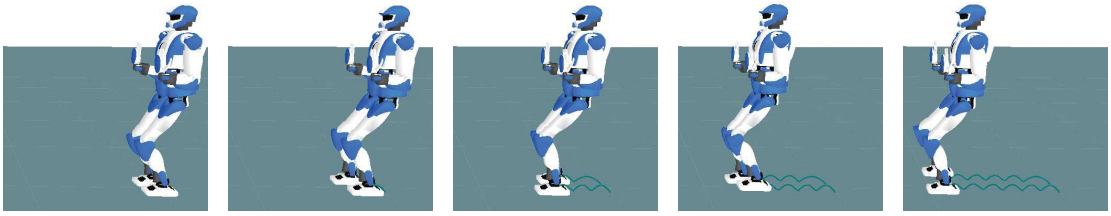


Figure 4.12: Image sequence of walking while holding a front-wrap body grasp.

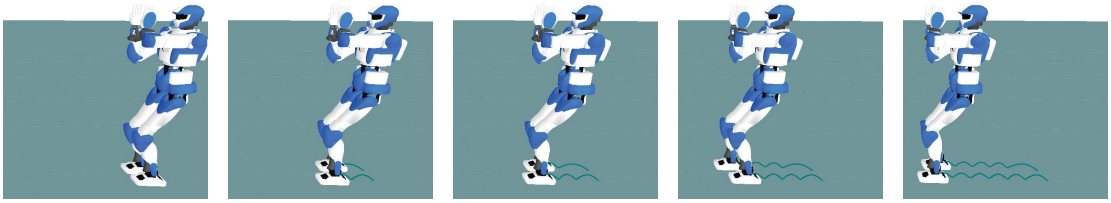


Figure 4.13: Image sequence of walking while holding a shoulder-mount body grasp.

Figure 4.14: Screenshots of experiments with the HRP-4 robot. Left: shoulder-mount body grasp while walking, Right: front-wrap body grasp while walking

Conclusion

This thesis explores several key aspects of a simple yet substantial idea: humanoid robots that have the ability to collaborate with humans. We start by exploring how visual information can be used in the context of haptic joint actions. For this, we implemented a framework using visual servoing and impedance control for doing the joint actions. Proactivity was achieved thanks to visual servoing, which allows the robot to have its own perception of the state of the task, thus to act on its own, even without sensing the human intention. Nevertheless, the human intention still needed to be accounted for. This was done with impedance control. This visio-haptic framework was implemented and tested on the HRP-2 humanoid robot in two example scenarios. Next, we revisited locomotion and balance in relation to physical interaction. For this, we designed walking pattern generators that not only take into account the physical interaction, but also use it accordingly, to operate as a follower or leader. Finally, we looked at the task of collaborative carrying as a whole. To do this, we started by creating a taxonomy based on the observation of several cases of human teams. We then try to infer from this the core principles of collaborative carrying, in order to program it on a humanoid robot. This is done by first creating a generic Finite State Machine, that encompasses all collaborative carrying tasks. This is then used to design objectives and constraints of an optimization problem that allows us to control the whole body of the humanoid robot.

Although the work here already presents a holistic approach to collaborating humanoids, there is much to be done as a whole and even in each specific area. We envision humanoid hardware, along with sensing, to continue improving. In particular, our work can greatly benefit from improvements on vision and haptic sensing and processing. In particular, to better leverage the continuous advancements in algorithms for computer vision and haptics, we would greatly benefit from having a robotic skin instead of localized force/torque sensing. Advances in computing might also enable the use of algorithms that right now are considered too slow to be useful. On the other hand, better actuation would allow to try for more ambitious experiments and test cases.

Although we can rely on other areas to progress, we must also strive to progress in parallel. As for the continuation of this work here specifically, we can envision various short, medium and long term improvements for each of the aspects we worked on. In general, for the short term, there are a lot of ideas left to be tested and demonstrations that can be done. Some demonstrations were left undone due to

lack of time. However, the algorithms presented are generic enough to be tested on other collaborative carrying cases. What follows are the ideas for improvements, specific to each of the areas of our work here.

For the work done in chapter 2 concerning vision and haptic sensing in collaboration, there are a lot of paths to be taken. In the short term, an important idea can be tested in the framework described. It was briefly mentioned that the impedance gains may serve to weigh between robot intention and complying to human intention. This can be better represented by adapting the gains, according to certainty of the sensing. For example, in the ball on table case study, if the visual tracking is lost, then the robot must rely completely on human intention through haptics. On the contrary, if the human intention is wavering and unsure, and if the visual tracking has a confident estimate, then the robot must try to do the task according to its own intention. This also leads to more studies that will need to be done on role switching and on other aspects of cognitive and behavioral science, related to human-robot collaboration. In the mid term, we need to explore various other possibilities of utilizing both vision and haptics information together. Our work has focused on how the information can be used in a control framework. We envision that some level of information fusion in sensor data processing will be necessary. On the contrary, using the information sources differently, in high-level decision making and planning aspects, is also possible. In fact, all of these methods can co-exist with each together. However, this path for continuation will need to heavily leverage new sensors and sensor processing methods, which would extend it to a long-term plan.

Chapter 3 concentrated on the locomotion and balance aspects of humanoid robots. In particular, we presented specific WPGs for the leader and follower cases. For the short term, one key issue we saw from real experiments, is the need for a high-fidelity force control in the leader WPG, since the interaction is used to balance the robot. As for the follower WPG, we only briefly mentioned the wrench prediction model for better proactive behaviors. In this work, the wrench is simply predicted to be constant throughout the preview horizon. Better prediction models may result in better proactive behaviors. Apart from improving the WPG itself, one envisioned mid term goal is a better integration into whole-body control, with work such as [87]. Another mid term goal is to try this framework in other physical interaction cases. Since the abstraction was made generic enough, the core idea is not specific to collaboration. Some other WPG objectives may be designed for push recovery, or simply in interacting with the environment in multi-contact situations. For the long term, we envision that the WPG will be well integrated into whole-body control and both of these aspects will be treated together. However to do this, we would need to handle several issues and explore several methods. For example, the use of more general but slower non-linear optimization frameworks may lead to better results.

Lastly, chapter 4 focused on the collaborative carrying task and its implementation within a whole-body control framework. In the short term, more demonstrations can be done along with a possible extension of the framework to other haptic joint

actions. In the mid term, there is a lot to be done for whole-body control. Firstly, we have briefly covered the different design choices and trade-offs for optimization. These might need to be considered more carefully, taking into account a broader application. Furthermore, various improvements can be made in the objective function and constraints design. Similarly to what we presented, other established approaches from control theory might benefit from a direct translation into optimization. For example, visual servoing and impedance control may be better integrated directly as tasks of the optimization. For the long term, we need to continuously move towards more general approaches, and remove the simplifications that are done to enable today's demonstrations. These include, for example, the *no-slipping* contact, and rigid-body, assumptions. Indeed, the use of vision, haptics, other sensor information, locomotion and balance must all be better integrated for a true *whole-body* approach.

Finally, for a broader perspective, we have to rethink humanoids and what they represent. Certainly, usability and practical future applications are one aspect. However, humanoids can also be a tool to learn about ourself, similar to our studies on collaboration. Better technology can open much broader topics and studies for the far future.

Appendix A

Condensing a model predictive control problem

Condensing is the elimination of future states and is commonly used in a Model Predictive Control (MPC) Problem. Consider a linear model of the following form:

$$\begin{aligned} \mathbf{x}_{k+1} =& \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k, \\ \mathbf{y}_{k+1} =& \mathbf{D}_k \mathbf{x}_k + \mathbf{E}_k \mathbf{u}_k + \mathbf{F}_{k+1} \mathbf{f}_{k+1}, \end{aligned} \tag{A.1}$$

where the discrete time interval k spans from $k = 0$ to $k = N$, with N being the length of the preview (prediction) horizon. Vectors $\mathbf{x}_k$, $\mathbf{y}_k$, and $\mathbf{u}_k$ are the k-th state, output and control input respectively. Note that the output may have a different form for other problems, but this form has been chosen here, because of its similarity to Eq.(3.12). Naively expanding the state equation of (A.1) to express all the previewed states results in:

$$\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_1 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_{N-1} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_1 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}. \tag{A.2}$$

Instead of this, the future states can be eliminated or *condensed*. This amounts to finding the matrices $\mathbf{U}_x$ and $\mathbf{U}_u$ such that:

$$\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \mathbf{U}_x \mathbf{x}_0 + \mathbf{U}_u \begin{bmatrix} \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}. \tag{A.3}$$

Each row of these matrices can be found by recursively applying Eq.(A.1), until the result is only a function of the current state $\mathbf{x}_0$. The resulting matrices are:

$$\mathbf{U}_\mathrm{x} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1\mathbf{A}_0 \\ \vdots \\ \mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_0 \end{bmatrix}, \quad \mathbf{U}_\mathrm{u} = \begin{bmatrix} \mathbf{B}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_1\mathbf{B}_0 & \mathbf{B}_1 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_1\mathbf{B}_0 & \mathbf{A}_{\mathrm{N\text{-}1}} \dots \mathbf{A}_2\mathbf{B}_1 & \dots & \mathbf{B}_{\mathrm{N\text{-}1}} \end{bmatrix}. \tag{A.4}$$

The same can be done for the output, so that:

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_\mathrm{N} \end{bmatrix} = \mathbf{O}_\mathrm{x}\mathbf{x}_0 + \mathbf{O}_\mathrm{u} \begin{bmatrix} \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{\mathrm{N\text{-}1}} \end{bmatrix} + \mathbf{O}_\mathrm{f} \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_\mathrm{N} \end{bmatrix}, \tag{A.5}$$

with:

$$\begin{aligned} \mathbf{O}_\mathrm{x} &= \begin{bmatrix} \mathbf{D}_0 \\ \mathbf{D}_1\mathbf{A}_0 \\ \mathbf{D}_2\mathbf{A}_1\mathbf{A}_0 \\ \vdots \\ \mathbf{D}_{\mathrm{N\text{-}1}}\mathbf{A}_{\mathrm{N\text{-}2}} \dots \mathbf{A}_0 \end{bmatrix}, \\ \mathbf{O}_\mathrm{u} &= \begin{bmatrix} \mathbf{E}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{D}_1\mathbf{B}_0 & \mathbf{E}_1 & \dots & \mathbf{0} \\ \mathbf{D}_2\mathbf{A}_1\mathbf{B}_0 & \mathbf{D}_2\mathbf{B}_1 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_{\mathrm{N\text{-}1}}\mathbf{A}_{\mathrm{N\text{-}2}} \dots \mathbf{A}_1\mathbf{B}_0 & \mathbf{D}_{\mathrm{N\text{-}1}}\mathbf{A}_{\mathrm{N\text{-}2}} \dots \mathbf{A}_2\mathbf{B}_1 & \dots & \mathbf{E}_{\mathrm{N\text{-}1}} \end{bmatrix}, \\ \mathbf{O}_\mathrm{f} &= \begin{bmatrix} \mathbf{F}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{F}_\mathrm{N} \end{bmatrix}. \end{aligned} \tag{A.6}$$

Notice also that, by first constructing $\mathbf{U}_\mathrm{x}$ and $\mathbf{U}_\mathrm{u}$, part of their structure can be reused to construct $\mathbf{O}_\mathrm{x}$ and $\mathbf{O}_\mathrm{u}$, respectively.

Bibliography

[1] N. Jarrassé, T. Charalambous, and E. Burdet, "A framework to describe, analyze and generate interactive motor behaviors," *PloS one*, vol. 7, no. 11, p. e49945, 2012.

[2] K. Reed and M. Peshkin, "Physical Collaboration of Human-Human and Human-Robot Teams," *Haptics, IEEE Transactions on*, vol. 1, pp. 108–120, July 2008.

[3] A. Kheddar, "Human-robot haptic joint actions is an equal control-sharing approach possible?," in *4th International Conference on Human System Interactions (HSI)*, pp. 268–273, IEEE, 2011.

[4] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *The journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[5] M. O. Ernst and H. H. Bülthoff, "Merging the senses into a robust percept," *Trends in cognitive sciences*, vol. 8, no. 4, pp. 162–169, 2004.

[6] M. O. Ernst and M. S. Banks, "Humans integrate visual and haptic information in a statistically optimal fashion," *Nature*, vol. 415, no. 6870, pp. 429–433, 2002.

[7] "Robots and robotic devices."
http://www.iso.org/iso/iso_technical_committee%3Fcommid%3D54138.

[8] Y. Maeda, T. Hara, and T. Arai, "Human-robot cooperative manipulation with motion estimation," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, vol. 4, pp. 2240–2245, 2001.

[9] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in *IEEE International Conference on Robotics and Automation*, pp. 4326–4332, 2011.

[10] A. Bussy, A. Kheddar, A. Crosnier, and F. Keith, "Human-humanoid haptic joint object transportation case study," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 3633–3638, 2012.

[11] L. Villani and J. De Schutter, "Force Control," in *Springer handbook of robotics* (B. Siciliano and O. Khatib, eds.), ch. 7, pp. 161–185, Springer, 2008.

[12] N. Hogan, "Impedance control - An approach to manipulation. I - Theory. II - Implementation. III - Applications," *ASME Transactions Journal of Dynamic Systems and Measurement Control B*, vol. 107, pp. 1–24, Mar. 1985.

[13] R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3, pp. 3097–3102 vol.3, May 1995.

[14] R. Ikeura, T. Moriguchi, and K. Mizutani, "Optimal variable impedance control for a robot and its application to lifting an object with a human," in *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pp. 500–505, 2002.

[15] V. Duchaine and C. M. Gosselin, "General model of human-robot cooperation using a novel velocity based variable impedance control," in *EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 446–451, IEEE, 2007.

[16] B. Corteville, E. Aertbeliën, H. Bruyninckx, J. De Schutter, and H. Van Brussel, "Human-inspired robot assistant for fast point-to-point movements," in *IEEE International Conference on Robotics and Automation*, pp. 3639–3644, 2007.

[17] E. Burdet, R. Osu, D. W. Franklin, T. E. Milner, and M. Kawato, "The central nervous system stabilizes unstable dynamics by learning optimal impedance," *Nature*, vol. 414, no. 6862, pp. 446–449, 2001.

[18] B. J. Nelson, J. D. Morrow, and P. K. Khosla, "Improved force control through visual servoing," in *Proc. of the American Control Conference*, vol. 1, pp. 380–386, IEEE, 1995.

[19] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981.

[20] J. Baeten, H. Bruyninckx, and J. De Schutter, "Integrated vision/force robotic servoing in the task frame formalism," *International Journal of Robotics Research*, vol. 22, no. 10-11, pp. 941–954, 2003.

[21] K. Hosoda, K. Igarashi, and M. Asada, "Adaptive hybrid visual servoing/force control in unknown environment," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, vol. 3, pp. 1097–1103 vol.3, 1996.

[22] M. Prats, P. Martinet, A. P. del Pobil, and S. Lee, "Vision force control in task-oriented grasping and manipulation," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 1320–1325, 2007.

[23] A. De Santis, V. Lippiello, B. Siciliano, and L. Villani, "Human-robot interaction control using force and vision," *Advances in Control Theory and Applications*, pp. 51–70, 2007.

[24] G. Morel, E. Malis, and S. Boudet, "Impedance based combination of visual and force control," in *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1743–1748, 1998.

[25] E. Malis, G. Morel, and F. Chaumette, "Robot control using disparate multiple sensors," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 364–377, 2001.

[26] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter, "iTASC: a tool for multi-sensor integration in robot manipulation," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 426–433, 2008.

[27] G. Taylor and L. Kleeman, *Visual Perception and Robotic Manipulation: 3D Object Recognition, Tracking and Hand-Eye Coordination.* Springer Tracts in Advanced Robotics, Springer, 2006.

[28] S. Wieland, D. Gonzalez-Aguirre, N. Vahrenkamp, T. Asfour, and R. Dillmann, "Combining force and visual feedback for physical interaction tasks in humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 439–446, 2009.

[29] B. J. Nelson and P. K. Khosla, "Force and vision resolvability for assimilating disparate sensory feedback," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 714–731, 1996.

[30] R. Featherstone, *Rigid body dynamics algorithms.* Springer, 2008.

[31] Z. Wang, K. Mülling, M. P. Deisenroth, H. Ben Amor, D. Vogt, B. Schölkopf, and J. Peters, "Probabilistic movement modeling for intention inference in human–robot interaction," *International Journal of Robotics Research*, vol. 32, no. 7, pp. 841–858, 2013.

[32] D. Song, N. Kyriazis, I. Oikonomidis, C. Papazov, A. Argyros, D. Burschka, and D. Kragic, "Predicting human intention in visual observations of hand/object interactions," in *IEEE International Conference on Robotics and Automation*, pp. 1608–1615, 2013.

[33] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *International Conference on Advanced Robotics*, pp. 1–6, IEEE, 2009.

[34] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 4414–4419, Sept 2011.

[35] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, K. Kaneko, M. Morisawa, E. Yoshida, and F. Kanehiro, "Vertical ladder climbing by the HRP-2 humanoid robot," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pp. 671–676, Nov 2014.

[36] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, 2014.

[37] S. Kajita, H. Hirukawa, K. Harada, K. Yokoi, and S. Sakka, *Introduction à la commande des robots humanoïdes: De la modélisation à la génération du mouvement*. Springer, 2009.

[38] P.-B. Wieber, R. Tedrake, and S. Kuindersma, "Modeling and control of legged robots," in *Springer handbook of robotics* (B. Siciliano and O. Khatib, eds.), ch. 48, Springer, second ed., 2015.

[39] P.-B. Wieber, "Holonomy and Nonholonomy in the Dynamics of Articulated Motion," in *Fast Motions in Biomechanics and Robotics* (M. Diehl and K. Mombaur, eds.), vol. 340 of *Lecture Notes in Control and Information Sciences*, pp. 411–425, Springer Berlin Heidelberg, 2006.

[40] M. Vukobratović and B. Borovac, "Zero-Moment Point — Thirty Five Years Of Its Life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004.

[41] 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 *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1620–1626, 2003.

[42] S.-H. Lee and A. Goswami, "Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots," in *IEEE International Conference on Robotics and Automation*, pp. 4667–4672, April 2007.

[43] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa, "Dynamics and balance of a humanoid robot during manipulation tasks," *IEEE Transactions on Robotics*, vol. 22, pp. 568–575, June 2006.

[44] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa, "A universal stability criterion of the foot contact of legged robots - adios ZMP," in *IEEE International Conference on Robotics and Automation*, pp. 1976–1983, May 2006.

[45] T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models," *International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[46] J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid," *International Journal of Robotics Research*, vol. 31, no. 10, pp. 1117–1133, 2012.

[47] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[48] K. Yokoyama, H. Handa, T. Isozumi, Y. Fukase, K. Kaneko, F. Kanehiro, Y. Kawai, F. Tomita, and H. Hirukawa, "Cooperative works by a human and a humanoid robot," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2985–2991, 2003.

[49] A. Bussy, P. Gergondet, A. Kheddar, F. Keith, and A. Crosnier, "Proactive behavior of a humanoid robot in a haptic transportation task with a human partner," in *IEEE International Symposium on Robot and Human Interactive Communication*, pp. 962–967, 2012.

[50] D. J. Agravante, A. Cherubini, A. Bussy, P. Gergondet, and A. Kheddar, "Collaborative Human-Humanoid Carrying Using Vision and Haptic Sensing," in *IEEE International Conference on Robotics and Automation*, pp. 607–612, May 2014.

[51] K. Nishiwaki and S. Kagami, "Online walking control system for humanoids with short cycle pattern generation," *International Journal of Robotics Research*, vol. 28, no. 6, pp. 729–742, 2009.

[52] K. Harada, S. Kajita, F. Kanehiro, K. Fujiwara, K. Kaneko, K. Yokoi, and H. Hirukawa, "Real-time planning of humanoid robot's gait for force controlled manipulation," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 616–622, April 2004.

[53] S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Controlling the planar motion of a heavy object by pushing with a humanoid robot using dual-arm force control," in *IEEE International Conference on Robotics and Automation*, pp. 1428–1435, May 2012.

[54] S. Nozawa, I. Kumagai, Y. Kakiuchi, K. Okada, and M. Inaba, "Humanoid full-body controller adapting constraints in structured objects through updating task-level reference force," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 3417–3424, Oct 2012.

[55] T. Takubo, K. Inoue, K. Sakata, Y. Mae, and T. Arai, "Mobile manipulation of humanoid robots - control method for CoM position with external force," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, vol. 2, pp. 1180–1185, Sept 2004.

[56] T. Takubo, K. Inoue, and T. Arai, "Pushing an Object Considering the Hand Reflect Forces by Humanoid Robot in Dynamic Walking," in *IEEE International Conference on Robotics and Automation*, pp. 1706–1711, April 2005.

[57] M. Stilman, K. Nishiwaki, and S. Kagami, "Humanoid teleoperation for whole body manipulation," in *IEEE International Conference on Robotics and Automation*, pp. 3175–3180, May 2008.

[58] A. Ibanez, P. Bidaud, and V. Padois, "Unified preview control for humanoid postural stability and upper-limb interaction adaptation," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 1801–1808, Oct 2012.

[59] M. Murooka, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Whole-body pushing manipulation with contact posture planning of large and heavy object for humanoid robot," in *IEEE International Conference on Robotics and Automation*, pp. 5682–5689, May 2015.

[60] M. Morisawa, K. Harada, S. Kajita, K. Kaneko, J. Sola, E. Yoshida, N. Mansard, K. Yokoi, and J.-P. Laumond, "Reactive stepping to prevent falling for humanoids," in *humanoids*, pp. 528–534, Dec 2009.

[61] M. Morisawa, F. Kanehiro, K. Kaneko, N. Mansard, J. Sola, E. Yoshida, K. Yokoi, and J.-P. Laumond, "Combining suppression of the disturbance and reactive stepping for recovering balance," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 3150–3156, Oct 2010.

[62] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online decision of foot placement using singular LQ preview regulation," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 13–18, Oct 2011.

[63] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online walking pattern generation for push recovery and minimum delay to commanded change of direction and speed," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 3411–3416, Oct 2012.

[64] K. Kosuge, M. Sato, and N. Kazamura, "Mobile robot helper," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 583–588 vol.1, 2000.

[65] J. Stückler and S. Behnke, "Following human guidance to cooperatively carry a large object," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 218–223, 2011.

[66] E. Berger, D. Vogt, N. Haji-Ghassemi, B. Jung, and H. B. Amor, "Inferring guidance information in cooperative human-robot tasks," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[67] M. Bellaccini, L. Lanari, A. Paolillo, and M. Vendittelli, "Manual guidance of humanoid robots without force sensors: Preliminary experiments with NAO," in *IEEE International Conference on Robotics and Automation*, pp. 1184–1189, May 2014.

[68] S. McGill and D. Lee, "Cooperative humanoid stretcher manipulation and locomotion," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 429–433, 2011.

[69] P. Evrard, *Contrôle d'humanoïdes pour réaliser des tâches haptiques en coopération avec un opérateur humain*. PhD thesis, Université de Montpellier 2, Montpellier, France, 2009.

[70] P. Evrard and A. Kheddar, "Homotopy switching model for dyad haptic interaction in physical collaborative tasks," in *EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 45–50, IEEE, 2009.

[71] A. Bussy, *Cognitive approach for representing the haptic physical human-humanoid interaction*. PhD thesis, Université de Montpellier 2, Montpellier, France, 2013.

[72] F. Chaumette and S. Hutchinson, "Visual servo control, Part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[73] F. Chaumette and S. Hutchinson, "Visual servo control, Part II: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[74] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital image processing using MATLAB*, vol. 2. Gatesmark Publishing Tennessee, 2009.

[75] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[76] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Incorporated, 2008.

[77] V. Lepetit and P. Fua, *Monocular model-based 3d tracking of rigid objects: A survey*. Now Publishers Inc, 2005.

[78] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, "Real-time markerless tracking for augmented reality: the virtual visual servoing framework," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 615–628, 2006.

[79] É. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.

[80] C. Dune, A. Herdt, E. Marchand, O. Stasse, P.-B. Wieber, and E. Yoshida, "Vision based control for humanoid robots," in *IROS Workshop on Visual Control of Mobile Robots (ViCoMoR)*, pp. 19–26, 2011.

[81] S. Gay, A. Ijspeert, and J. S. Victor, "Predictive gaze stabilization during periodic locomotion based on Adaptive Frequency Oscillators," in *IEEE International Conference on Robotics and Automation*, pp. 271–278, 2012.

[82] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear control via approximate input-output linearization: The ball and beam example," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 392–398, 1992.

[83] H. Audren, J. Vaillant, A. Kheddar, A. Escande, K. Kaneko, and E. Yoshida, "Model preview control in multi-contact motion-application to a humanoid robot," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 4030–4035, Sept 2014.

[84] W. Chung, L.-C. Fu, and S.-H. Hsu, "Motion Control," in *Springer handbook of robotics* (B. Siciliano and O. Khatib, eds.), ch. 6, pp. 133–159, Springer, 2008.

[85] D. J. Agravante, A. Cherubini, A. Bussy, and A. Kheddar, "Human-humanoid joint haptic table carrying task with height stabilization using vision," in *IEEE/RSJ International Conference on Robots and Intelligent Systems*, pp. 4609–4614, 2013.

[86] D. J. Agravante, A. Cherubini, and A. Kheddar, "Using vision and haptic sensing for human-humanoid joint actions," in *IEEE International Conference on Robotics, Automation and Mechatronics* , pp. 13–18, 2013.

[87] A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Whole body motion controller with long-term balance constraints," in *IEEE-RAS International Conference on Humanoid Robots*, pp. 444–450, Nov 2014.

[88] M. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 269–279, Jun 1989.

[89] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *International Journal of Robotics Research*, 2012.

[90] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, Springer New York, 2000.

[91] J. Vaillant, *Programmation de Mouvements de Locomotion et Manipulation en Multi-Contacts pour Robots Humanoïdes et Expérimentations*. PhD thesis, Université de Montpellier 2, Montpellier, France, 2015.

Abstract
Humanoid robots provide many advantages when working together with humans to perform various tasks. Since humans in general have a lot of experience in physically collaborating with each other, a humanoid with a similar range of motion and sensing has the potential to do the same. This thesis is focused on enabling humanoids that can do such tasks together with humans: collaborative humanoids. In particular, we use the example where a humanoid and a human collaboratively carry and transport objects together. However, there is much to be done in order to achieve this. Here, we first focus on utilizing vision and haptic information together for enabling better collaboration. More specifically the use of vision-based control together with admittance control is tested as a framework for enabling the humanoid to better collaborate by having its own notion of the task. Next, we detail how walking pattern generators can be designed taking into account physical collaboration. For this, we create leader and follower type walking pattern generators. Finally, the task of collaboratively carrying an object together with a human is broken down and implemented within an optimization-based whole-body control framework.

Résumé
Les robots humanoïdes sont les plus appropriés pour travailler en coopération avec l'homme. En effet, puisque les humains sont naturellement habitués à collaborer entre eux, un robot avec des capacités sensorielles et de locomotion semblables aux leurs, sera le plus adapté. Cette thèse vise à rendre les robot humanoïdes capables d'aider l'homme, afin de concevoir des 'humanoïdes collaboratifs'. On considère ici la tâche de transport collaboratif d'objets. D'abord, on montre comment l'utilisation simultanée de vision et de données haptiques peut améliorer la collaboration. Une stratégie combinant asservissement visuel et commande en admittance est proposée, puis validée dans un scénario de transport collaboratif homme/humanoïde. Ensuite, on présente un algorithme de génération de marche, prenant intrinsèquement en compte la collaboration physique. Cet algorithme peut être spécifié suivant que le robot guide (leader) ou soit guidé (follower) lors de la tâche. Enfin, on montre comment le transport collaboratif d'objets peut être réalisé dans le cadre d'un schéma de commande optimale pour le corps complet.