

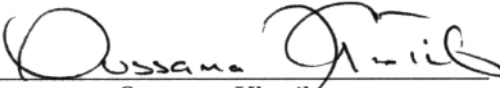
SYNTHESIS AND CONTROL OF WHOLE-BODY BEHAVIORS IN
HUMANOID SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Luis Sentis
July 2007

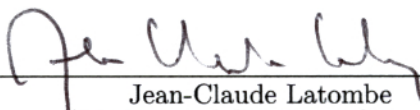
© Copyright by Luis Sentis 2007
All Rights Reserved

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.




Oussama Khatib
(Professor of Computer Science)
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



Jean-Claude Latombe
(Professor of Computer Science)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



Stephen Boyd
(Professor of Electrical Engineering)

Approved for the University Committee on Graduate Studies.

Abstract

A great challenge for robotic systems is their ability to carry on complex manipulation and locomotion tasks while responding to the changing environment. To allow robots to operate in human environments there is a strong need to develop new control architectures that can provide advanced task capabilities and interactive skills. These architectures must be effective in coordinating whole-body behaviors for various control objectives while complying with balance stability, contact stance, and other dynamic constraints. In addition, to facilitate the integration of robots in human environments, it is desirable for their motions and task behaviors to be compatible with those of humans. In this thesis, we present a control methodology for the synthesis of realtime whole-body control behaviors in humanoid systems. The work is presented in three parts.

First, we establish mathematical foundations that characterize the kinematic and dynamic behaviors of task and postural criteria under balance and contact stability constraints. We identify the dynamic behavior of postural tasks operating in the null space of operational tasks and we develop task-oriented controllers in postural space. These controllers are used to accomplish secondary goals or to optimize postural criteria without affecting priority tasks. Based on task and posture control decompositions we define recursive structures with multiple priority levels. These structures allow us to create controllers for all aspects of motion while ensuring that critical tasks are accomplished first. Exploiting prioritization, we address the control of dynamic constraints as priority tasks and we project operational tasks and postural criteria in the null space of all acting constraints. This strategy prevents lower priority tasks from violating the acting constraints.

Second, we develop a variety of controllers to address the different aspects of the robot's motion. We propose position and force controllers to control the various task effectors of the robot. We use potential fields to handle reactively dynamic constraints such as balance stability, joint limits, obstacle avoidance, and self collisions. We develop posture controllers to enhance overall performance in terms of available workspace, resemblance to

human poses, and optimization of actuation effort. Third, we tackle the synthesis of complex whole-body behaviors. To facilitate the creation of behaviors we develop control and behavioral abstractions that encapsulate behavior representation and action mechanisms. These abstractions are designed to be instantiated and coordinated by high level decision and perceptual processes.

The methods proposed throughout this dissertation provide numerous control tools and behavioral abstractions that will allow robots to deal effectively with the complexity and dynamism of human environments.

Acknowledgements

I would like to first thank my adviser and friend Oussama Khatib. Not only I had excellent tutoring from him but I enjoyed many interesting conversations over good sushi and espresso. Thanks to Jean Claude Latombe who invited me to present my control system when I first developed it and gave me the opportunity to discuss it. Thanks to Stephen Boyd whose courses on convex optimization taught me how to formulate new ideas using mathematical tools. Thanks to Scott Delp for introducing me to biomechanics and giving me feedback on my research. Thanks to Claire Tomlin whose courses on control theory gave me a strong research background. Thanks to Santiago Silvestre, my old time adviser at the Polytechnic University of Catalonia for his support on entering the Ph.D. program. Thanks to David Cheriton whose input allowed me to design software abstractions for our robotic simulator. Thanks to our lab administrators Hoa Nguyen, Sarah Lee, and late Jutta McCormick. Thanks to Peche Turner, CS department manager at Stanford, for her help on administrative issues.

Thanks to my labmates Oliver Brock, Alan Bowling, Kyong Sok Chang, Francois Conti, Emel Demircan, Vincent Desapio, Sam Hamner, Bob Holmberg, Jin-sung Kwon, Stan Ley, Charity Lu, Jae-Heung Park, Irina Pashcenko, Anya Petrovskaya, Diego Ruspini, Dong-Jun Shin, Constantinos Stratelos, Peter Thaulaud, Philip Tsai, Sriram Viji, James Warren, and Mike Zinn, many of whom became good friends.

Thanks to Taizo Yoshikawa and Yuji Haikawa members of Honda Co. who gave me the perspective for the application of our control system in the real humanoid robot. Thanks to Shane Chang, Chief Scientist at Honda R&D in Mountain View for coordinating efforts between our lab at Stanford and Honda labs in Japan. Thanks to Hector Gonzalez-Banos for showing me the facilities at Honda R&D in Mountain View and for discussing research issues.

Thanks to Javier Minguez, Pierre Olivier Latour, Federico Barbagli, Tine Lefebvre, Vincent and Karine Padois, Nicolas Mansard, Irene Sardellitti, Steve Burion, Rui Cortesao, Edwardo Fukushima, Xiowei Ma, Laurence Meylan, Herman Bruyninckx, Benoit Dagon,

Nicolas Tournier, Clara Kim, Mai Nguyen, Elena Pacchierotti, Roland Philippsen, Kazuhito Yokoi, and Nicolas Turro who came to visit our lab and spent time working with me and discussing various research topics.

Thanks to Gordon Cheng, James Kuffner, Marko Popovic, Roy Featherstone, and Jorge Moraleda for their insights and support on research issues.

An old professor of mine once told me "finish any endeavor with a long list of friends". During these years at Stanford I have followed his advice. I would like to thank Stefan Harmeling, Dana Porrat, John Townend, Ulrich Barnhoefer, Ofer Levi, Onn Brandman, Mattan Erez, Matteo Slanina, Vincent Vanhoucke, Drago Anguelov, Alessandra Lanzara, Rachel Kolodny, Dominik Zumbuhl, Laurence Melloul, Esra Inan, Momo Waguri, Ana Moral, Laila Mattos, Alicia Meuret, Deborah Berebichez, Minja Trklja, Greg Jefferis, Mor Naaman, Sylvia Smulin, Cesar Sanchez, Atosa Ghadimi, Michael Lustig, Yuval Nov, Ofer Levi (at Ginzton labs), Rely Brandman, Ana Pedros, Joyce Noah, Oren Shneerson, Pablo Molinero, Roland Wolkovicz, Alex Smola, Sebastian Brion, and Nurit Jugent for their friendship. Thanks also to Kerstin Johnsson, Persefoni Kyritsi, Victor Araman, and Angel Victor de Miguel with whom I spent good times. Thanks to Radu and Tanya Auf Der Heyde, Ali Tore, Michal and Doron Gal, Martin and Tanja Mai, Thomas and Myraida Finkbeiner for their friendship. Thanks to Javier Cardona and Celine Monget, my lovely friends from whom I have received much affection during these years. Thanks to John Smolowe for his advice. Thanks to Priscila Spolyar for her friendship. Thanks to Daniel, Diana, and Dylan Canyon Damer-Malinowski for their friendship during recent years.

I never forget Luis Vallescar, Anna Viladas, Toni Escude, Alex Herrera, Monica Aubert, Rafa Prat, Sara Cabot, Theo de Andres, Teresa Bas, Lucas Carne, Susana Junyent, Tomas Dualde, Jose Maria Llauro, and Marcos Portabella.

During my Ph.D. years I have received much affection and support from my parents Maria Angeles Alvarez and Luis Sentis, my sisters Eva and Ana Sentis, and my nieces Paula and Arantxa Carasa. Thanks to Clara and Claudia Moller, Isabel Alvarez, Claudio Moller, Maria Rosa Alvarez and Silvia Prim for their love and support. Thanks to Alberto Aguirre and Maru Diamante for their love and support. Thanks to my parents in law Suzi and Burhan Ben-Ezra Kosekaya for their love and example on family issues. Thanks to Patricia and Bea Wolf, and Luis Alabart for their love.

At the beginning of my Ph.D., I had an eye on this pretty and interesting girl walking on campus. She is now part of my family and a strong support on many aspects of my life as well as in my career. This thesis is dedicated to my wife Adela Ben-Yakar-Sentis and our wonderful son Dekel for bringing me a lot of happiness.

Contents

Abstract	iv
Acknowledgements	vi
1 Introduction	1
1.1 Motivation and Background	1
1.1.1 Control of Humanoid Systems	2
1.1.2 Synthesis of Whole-Body Behaviors	4
1.2 Objectives and Approach	4
1.3 Chapter Summary	6
2 Basic Representations for Whole-Body Control	8
2.1 Representation of Multi-Legged Robots	10
2.1.1 Kinematic and Dynamic Models	10
2.1.2 Impact of Supporting Contacts	16
2.2 Operational Space Control	18
2.2.1 Task Kinematics Under Supporting Constraints	20
2.2.2 Task Dynamics and Control	26
2.2.3 Characterization of the Residual Task Redundancy	31
2.3 Control of Internal Forces	34
2.4 Examples	35
2.4.1 Balance Stability	35
2.4.2 COG Vertical Control	37
3 Prioritized Multi-Task Control	39
3.1 Representation of Whole-Body Behaviors	41
3.1.1 Multi-Task Decomposition	42

3.1.2	Constrained Kinematics	47
3.2	Control Structures	48
3.2.1	Prioritization Versus Aggregation	48
3.2.2	Prioritized Dynamics and Control	53
3.2.3	Recursive Redundancy	56
3.2.4	Task Feasibility	58
3.2.5	Overview of Control Strategies	61
3.3	Examples	63
3.3.1	Tracking the Contour of an Object	64
3.3.2	Walking with Posture Variations	67
3.3.3	Dynamic Walking: First Step	68
4	Posture Control	73
4.1	Posture Control Structure	75
4.2	Task Based Postures	76
4.2.1	Example: Posture Control to Avoid an Overhead Obstacle	80
4.3	Criteria Based Postures	81
4.3.1	Imitation of Human Poses	81
4.3.2	Example 1: Upright Posture Attractor	86
4.3.3	Example 2: Upright Posture Control with Contact Task	88
4.3.4	Example 3: Posture Behavior with Interactive Task	89
4.3.5	Example 4: Posture Switching To Optimize Reachable Workspace	89
4.3.6	Effort Minimization	92
4.3.7	Example 1: Effort Minimization while Standing Up	94
4.3.8	Example 2: Effort Minimization with Interactive Task	94
4.4	Posture Stiffness Control	97
4.4.1	Examples: Response to Force Perturbations	98
5	Realtime Handling of Dynamic Constraints	101
5.1	Control Structure to Handle Constraints	104
5.1.1	Constraint Prioritization	104
5.1.2	Realtime Response to Dynamic Constraints	108
5.1.3	Constraint-Consistent Task Control	110
5.1.4	Task Feasibility Under Constraints	112
5.2	Types of Constraints and Control Approaches	114
5.2.1	Support Constraints	115

5.2.2	Balance Constraints	117
5.2.3	Obstacle Avoidance	117
5.2.4	Joint Limit Constraints	120
5.2.5	Self Collision Avoidance	122
6	Whole-Body Control of Movements in Midair	127
6.1	Basic Equations of Motion and Physics of Free Space	129
6.1.1	Joint Space Dynamics	129
6.1.2	Analysis of the System's Momentum	131
6.1.3	Constrained Kinematics	133
6.2	Operational Space Control	135
6.2.1	Task Dynamics and Control	135
6.2.2	Task Feasibility	137
6.3	Prioritized Multi-Task Control	139
6.3.1	Representations and control structures	140
6.3.2	Task Feasibility	145
6.3.3	Posture Dynamics and Control	146
6.4	Examples	150
6.4.1	Forward Jumping	150
6.4.2	Kick in Mid Air	153
6.4.3	Twist'N'Jump	153
7	Realtime Synthesis of Whole-Body Behaviors	155
7.1	Composition and Instantiation of Low-Level Behaviors	157
7.1.1	Control Primitives	158
7.1.2	Task Creation	159
7.1.3	Task Execution	164
7.2	Composition and Instantiation of Whole-Body Behaviors	164
7.2.1	Action Primitives	166
7.2.2	Whole-Body Behaviors	167
7.2.3	Behavior Feasibility	168
8	Concluding Remarks	170
8.1	Summary of Results and Contributions	170
8.1.1	Whole-Body Control Methods	170
8.1.2	Reactive Handling of Dynamic Constraints	173

8.1.3	Synthesis of Whole-Body Behaviors	173
8.1.4	Implementation Details	173
8.2	Discussion	175
8.3	Summary of Publications	175
8.4	Future Work	176
A	Mathematical Proofs	178

List of Tables

3.1	Screwgun manipulation	44
3.2	Task decomposition for walking behavior	46
3.3	Prioritized hierarchy	50
6.1	Volleyball task decomposition	140
6.2	Task hierarchy	141
7.1	Library of control primitives	160

List of Figures

2.1	Multi-legged robots	10
2.2	Kinematic representation of a legged robot	12
2.3	Non-fixed branching representation	13
2.4	Reaction forces on the feet	14
2.5	Whole-body walking behavior	19
2.6	Dependency between base and joint displacements	22
2.7	Kinematic singularities due to support constraints	26
2.8	Contact forces during task interactions	27
2.9	Control of static balance	36
2.10	Control of the COG's height	38
3.1	Manipulation behavior with screwgun	43
3.2	Simultaneous locomotion and manipulation behavior	45
3.3	Conflict between task goals	52
3.4	Task feasibility under prioritization	59
3.5	Condition number of prioritized inertias	60
3.6	Tracking the contour of an object	65
3.7	Gaze control	66
3.8	Posture attractors	67
3.9	Walking behavior wit posture variations	68
3.10	Walking phases	69
3.11	Torques during walking motion	70
3.12	Dynamic walking (first step)	72
4.1	Postural DOFs	76
4.2	Goal-based posture control	78
4.3	Posture control for avoiding overhead obstacle	80

4.4	Captured human poses	82
4.5	Upright pose attractor	87
4.6	Upright pose under contact	88
4.7	Posture behavior with interactive task	90
4.8	Posture switching	91
4.9	Human effort	92
4.10	Effort minimization while standing up	95
4.11	Effort minimization under hand control	96
4.12	Saturated joint velocities during effort minimization	97
4.13	Posture stiffness control	99
5.1	Joint limits control concept	105
5.2	Handling of joint limits on a humanoid	106
5.3	Constraint handling potentials	108
5.4	Example on task feasibility	113
5.5	Constrained behavior with foot on pedestal	116
5.6	Obstacle avoidance concept	118
5.7	Example on collision avoidance	119
5.8	Task feasibility under collision avoidance	121
5.9	Example involving multiple joint limit constraints	123
5.10	Example involving head orientation under joint limits	124
5.11	Example on self collision avoidance	126
6.1	Straddle jump	128
6.2	Task feasibility in free space	138
6.3	Multi task free space behavior	139
6.4	Forward jumping behavior	150
6.5	Torque values during jumping	152
6.6	Jumping forward with kick	153
6.7	Twist 'n' jump behavior	154
7.1	Connection between behavioral and execution layers	157
7.2	Execution layer	158
7.3	Control primitives	159
7.4	Task decomposition	161
7.5	Robot model	162

7.6	Relative importance of task categories	163
7.7	Operation of the movement layer	165
7.8	Instantiation of movements	167
7.9	Action sequencing	168
7.10	Handling of infeasible tasks	169

Chapter 1

Introduction

A humanoid is a multipurpose robot designed to perform human-like manipulation and locomotion tasks aiming to assist the human. Let us control it! The goal of this dissertation is to develop a methodology for the control of humanoid robots in dynamic environment.

1.1 Motivation and Background

Humanoids could one day become our personal helpers extending human capabilities in a variety of fields. Anthropomorphism allows humanoids to share our same environments and maybe soon do similar chores than we do. However, high mechanical complexity as well as limitations on control and perceptual capabilities prevents their further application into our environments.

In a near future, humanoids should be able to perform human-level tasks in dynamic environments. They should be able to manipulate objects, move around, and respond to the environment in similar ways than humans do or even better. Through cooperation, they should be able to perform sophisticated tasks such as building large structures or operating complex machinery.

The characterization and control of humanoid systems has an impact beyond robotics. It can provide the support to understand biological functions of the human body (biomechanics), tools to design machines and spaces where humans operate (ergonomics), simulation environments to study the effects of musculoskeletal alterations (surgical simulation) and to design and study rehabilitation systems, and support to synthesize realistic computer animations.

This dissertation will explore control issues in humanoid systems and methods for the real-time synthesis of whole-body behaviors, aiming to extend the frontiers of robotics research

and related fields.

1.1.1 Control of Humanoid Systems

Techniques for the control of humanoid systems derive from methods originally developed for industrial manipulators, mostly based on inverse kinematic techniques. However, inverse kinematic techniques pose important limitations because they require the generation of joint trajectories hindering contact interactions and complicating balance control among some problems. In response to these limitations, we will develop here a whole-body control framework implementing torque control strategies for advanced contact and non-contact interactions.

We will also address the development of control strategies to respond to the changing environment. Currently, this is an open research area where little progress has been done when addressing humanoid systems. One of the reasons is because responding quickly to dynamic events while maintaining balance and contact stability needs to be addressed in the force domain instead of relying on trajectory generation. Realtime response to dynamic constraints will occupy one chapter of this dissertation. In the 1980s dynamic control strategies for the control of robotic manipulators were first developed (Khatib 1980; Khatib 1987). The control framework that we will develop throughout this dissertation will be largely based on these techniques.

Handling physical constraints during movement execution has received much attention over the past years, with most of the work focused on collision free movement for manipulator control and mobile navigation. Although we will discuss in detail novel collision avoidance techniques for humanoid systems we will also focus on new techniques to respond in realtime to a variety of internal and external constraints.

Collision free movement has been addressed both in the context of reactive control and in the context of motion planning (Pieper 1968; Nilsson 1969; Udupa 1977; Khatib and Maitre 1978). In (Khatib 1986), a potential field approach was proposed. Potential field techniques will be extensively used and extended throughout this dissertation to handle motion constraints in humanoid systems. A short list of relevant work on reactive techniques include (Liegeois 1977; Maciejewski and Klein 1985; Khatib 1986; Brooks 1986; Siciliano and Slotine 1991; Kwon et al. 1991; Espiau et al. 1992; Marchand and Hager 1998; Sentis and Khatib 2006). Work on relaxation of precomputed paths, one of the possible applications of reactive control, can be found in (Krogh 1984; Buckley 1986; Quinlan 1994; Brock et al. 2002). Motion planning for robot navigation, manipulation, and more recently whole-body control has received much attention with some of the most important work found in (Nilsson

1969; Udupa 1977; Moravec 1980; Chatila 1981; Latombe 1991; Laumond and Jacobs 1994; Lozano-Perez 1981; Kuffner et al. 2003; Minguez and Montano 2004; Hauser et al. 2006).

Humanoid systems differ from fixed-base or mobile manipulators in that they are supported by the ground and need to maintain balance stability. To characterize contact constraints we represent humanoids as free floating systems with six passive DOFs attached to their base. The action of the gravity causes reaction forces to appear on the robot's feet or on the supporting structures. Once in contact, the robot's motion is similar to that of parallel structures (Stewart 1965) and multigrasp systems (Kerr and Roth 1986; Cole, Hauser, and Sastry 1989). We will use some of the concepts developed for parallel systems to control humanoids in contact with the ground. To study the response of humanoids under supporting contacts we will characterize contact dynamics and establish dynamic constraints at contact points. Much work has been focused on the control of underactuated systems especially in the context of space robotics. Some of this work can be found in (Arai and Tachi 1991; Umetami and Yoshida 1989; Dubowsky and Papadopoulos 1993; Jain and Rodriguez 1993). We will combine concepts from contact dynamics and underactuated systems to develop operational space control strategies for humanoids under contact and balance constraints. Complementary control structures for movements in mid air will also be developed.

Although balance control has been extensively studied in multilegged systems (Raibert 1986; Hirai et al. 1998; Harada et al. 2004), we will propose here controllers that provide direct manipulation of COG accelerations, providing a flexible framework to implement static and dynamic balancing strategies.

To create complex behaviors, humanoids need to simultaneously accomplish multiple control objectives. For instance, locomotion, manipulation, balance, and posture stance, will need to be simultaneously controlled. We will dedicate one chapter to the development of a multi-task control framework. To decouple the control of high priority tasks from the control of lower priority tasks and to avoid conflicts between tasks or with the acting constraints we will develop a prioritized control approach that will ensure that high priority tasks are first accomplished and will provide the support to measure task feasibility at runtime. Prioritization was first proposed by (Hanafusa, Yoshikawa, and Nakamura 1981) in the context of inverse kinematics control. In this dissertation we will extend prioritization to operational space control. We will also characterize task feasibility under prioritized control. Previous work on multi-task control include (Nakamura et al. 1987; Siciliano and Slotine 1991). More recently, in (Baerlocher and Boulic 1998) a multitask control approach was developed for inverse kinematic control of computer generated characters.

1.1.2 Synthesis of Whole-Body Behaviors

It is our objective to provide control methods and supporting entities for the synthesis of autonomous behaviors in human environments. In this context, we have designed behavioral entities that could serve as the main units of action connecting to high level controllers. These entities encapsulate task decomposition and movement sequencing, abstracting the desired actions. For instance, we have designed behavioral abstractions that can be used to implement goal-oriented walking, manipulation, or jumping behaviors. Instead of relying on preprogrammed motions, our behavioral entities are designed to execute goals at runtime issued by perception systems (e.g. Petrovskaya, Park, and Khatib 2007) or teleoperation devices (e.g. Conti and Khatib 2005). Moreover, our entities are designed to monitor and respond to dynamic events such as contact interactions or moving obstacles.

Synthesizing whole-body behaviors in humanoid systems requires the coordination of multiple low-level tasks. To support the instantiation and coordination of tasks our behavioral abstractions encapsulate desired action mechanisms. These abstractions are designed to connect with high level controllers and decision systems that feed task goals and trigger the sequencing between movement states.

Related work on behavior synthesis include behavior-based control systems (Brooks 1986) and learning and imitation using movement primitives by (Arbib 1981; Billard 2000; Mataric 2002; Schaal et al. 2003).

1.2 Objectives and Approach

The objective of this dissertation is to develop control and behavioral methods that will serve as a platform to synthesize autonomous behaviors on humanoid systems for operations in human environments.

To synthesize complex behaviors in dynamic environments, our approach will consist on implementing multiple controllers to deal simultaneously with manipulation, locomotion, postural, and constraint handling tasks. Our controllers will implement operational space control strategies (Khatib 1987) unifying the control of motion and forces while accounting for the physical dynamics of the robotic system.

To characterize the overall mobility of humanoids in space we will represent them as branching structures with respect to a free floating base. The equations of motion of the overall system will be expressed in terms of these representations allowing controllers to coordinate whole-body motions to accomplish the desired task goals. Whole-body representations and control of branching mechanisms were previously discussed in (Russakov

et al. 1995). We will develop similar representations for our robotic systems that will allow controllers to simultaneously accomplish multiple task goals while characterizing the residual movement redundancy.

To avoid the computation of trajectories, our methods will rely on the implementation of potential fields. Potential fields can be mapped into control forces and subsequently into control torques, which can be directly sent to the actuators. Throughout this dissertation we will develop potential field techniques to control manipulation, locomotion, balance, and postural behaviors.

To provide the support for operations in dynamic environments we will develop control strategies to handle internal and external constraints in realtime. Contact constraints will be directly integrated within kinematic and dynamic representations, allowing all motions to be automatically compliant with supporting contacts. Balance, obstacle avoidance, self-collision avoidance, and joint limits will be handled reactively using potential fields. Reactive handling of constraints will allow humanoids to accomplish tasks in dynamically changing environments. Although we will focus on reactive techniques, our methods could be extended to path modification such as in elastic strips (Brock and Khatib 2002). To deal with dynamic constraints such as contacts, joint limits, and moving obstacles, we will develop prioritized control strategies (Nakamura, Hanafusa, and Yoshikawa 1987) that will prevent operational tasks from violating the acting constraints. Our control framework will be design to respond to dynamic constraints without interrupting the global task. However, when constraints become too severe, manipulation and locomotion tasks may not be feasible. To deal with these situations we will design methods to measure task feasibility at runtime and modified the robot's behavior accordingly.

To control balance stability we will develop operational space methods to control COG accelerations providing the support to implement static or dynamic balance strategies (Vukobratovic and Borovac 2004; Harada et al. 2004).

Besides controlling operational tasks under contact and balance constraints, we will develop control methods for the control of postural behavior. Postural behavior is especially important in humanoids because they are highly mobile systems. We will characterize postural motion by identifying the motion manifold that is compliant with operational tasks and the acting constraints and we will use postures to optimize desired criteria or to track whole-body poses. In particular, we will describe control methods for pose imitation and effort minimization.

To support the creation of autonomous behaviors in complex environments we will develop behavioral abstractions that will encapsulate task decomposition and movement sequencing. These entities will be designed to implement complex movements with a minimal set of external parameters. In the future, these abstractions will be used in combination with perception and decision processes. We will develop task primitives that will describe task representations and control policies. Behavioral primitives will be created through the aggregation and coordination of sets of tasks. Instead of relying on trajectory generation, task primitives will be designed to accomplish arbitrary goals at runtime describing the motion or contact forces of the different parts of the robot's body.

The control framework we will describe throughout this dissertation is meant to control real humanoid robots. Although, we have used simulated humanoid models to validate our methods we are currently implementing our results into a real humanoid robot. We expect to show results soon.

1.3 Chapter Summary

This dissertation is organized as follows. In Chapter 2 we will present fundamental concepts for whole-body control of humanoid robots. We will first introduce a representation of multilegged robots as free floating branching systems and will characterize the effect of contact constraints. We will study the equations of motion of constrained systems and use them to develop operational space control strategies (Khatib 1987) to control arbitrary task objectives under supporting constraints.

In Chapter 3 we will develop control methods to simultaneous control multiple task points under balance and support constraints using operational space control strategies. To deal with conflicting scenarios between tasks, we will develop prioritized control techniques where lower priority tasks will operate in the residual redundancy of higher priority tasks. Task prioritization will prevent coupling of lower priority tasks into higher priority tasks and will provide support to determine task feasibility under the acting constraints.

In Chapter 4 we will characterize postural behavior under multi-task control. The residual space of motion will be used to enhance overall performance and to imitate captured human poses. Using postures we will be able to accomplish desired postural goals or optimize desired criteria without interrupting the global task.

In Chapter 5 we will present reactive methods to handle internal and external constraints. We will first develop novel control structures where operational and postural tasks will operate in the null space of all acting constraints. We will also develop constraint

handling tasks based on potential fields that will allow humanoids to respond in realtime to dynamic constraints.

In Chapter 6 we will extend whole-body control for behaviors in free space. We will characterize the equations of motion of free space focusing on the conservation of momenta. Using these representations we will develop multi-task controllers similar to the controllers develop from ground based behaviors. We will also analyze the feasibility of operational tasks and the control of postural tasks during movements in free space.

In Chapter 7 we will develop abstractions to support the synthesis of whole-body behaviors. We will first develop control abstractions for the instantiation and aggregation of operational tasks. Using these abstractions, we will develop behavioral entities that will encapsulate task decomposition and movement sequencing. These abstractions will be designed to connect to perception and decision processes with the aim to support the creation of emergent behaviors.

Chapter 2

Basic Representations for Whole-Body Control

In view of the structural differences between humanoid systems and fixed manipulators, the goal of this chapter is to develop kinematic, dynamic, and control representations that will serve as the basic elements for the development of a whole-body control framework. This components will be presented as an extension of the operational space control formulation (Khatib 1987) for humanoid systems. Operational space control will allow us to implement a variety of control strategies for effective interactions with the physical environment and for advanced control of balance stability.

One of the fundamental differences between humanoids and robotic manipulators is their detachment from the ground. Gravity forces push the humanoid's body against the ground, providing a supporting platform to move in all directions. The robot's movement is therefore not only determined by joint positions but also by the position and orientation of its body with respect to an inertial frame of reference. At the same time, reaction forces created between contact points and the ground are used for balance and locomotion. One of the goals of this chapter will be to characterize the mobility of humanoids in free space and the impact of supporting contacts at all levels.

A unique characteristic of humanoids is their high mobility. Humanoids are equipped with thirty or more moving joints allowing them to simultaneously coordinate manipulation, locomotion, and postural tasks while responding to the changing environments. This high mobility means that the mechanism is redundant with respect to operational tasks, or equivalently that operational tasks can be accomplished with different postural arrangements. To characterize the task's redundancy we will identify here the residual space of

motion (i.e. the posture space) that simultaneously complies with the desired tasks and the supporting contacts. In Chapter 4 we will use this characterization to enhance movement performance in posture space while executing complex tasks.

To control humanoid systems we need to simultaneously coordinate multiple operational tasks. There is a vast amount of work addressing the control of manipulation tasks on industrial manipulators. To control the robot's end effector, inverse kinematic solutions have been extensively used (Pieper 1968; Roth, Rastegar, and Sheinmann 1973). However, to engage in effective contact interactions dynamic control strategies were developed in the early 80s. Dynamic control was first proposed by (Khatib 1980) and subsequently extended into the operational space formulation (Khatib 1987). The main goal of this chapter will be to extend the operational space formulation to control operational tasks in humanoid systems.

An integral task during manipulation and locomotion behaviors is to maintain balance stability. The framework we will develop in this chapter will ensure that stable balance is being maintained and that internal forces between limbs in contact are properly controlled. We will study the kinematics and dynamics of the robot under contact constraints and propose operational space controllers to control balance as well as arbitrary task points. We will also discuss analogies between multi-legged systems, parallel systems, and underactuated systems. As a result, we will develop constraint-consistent mappings that will be used to project task space forces into actuator torques while complying with all acting contact constraints. Related work on constraint kinematics and dynamics can be found in (Fichter and McDowell 1980; Merlet 1996).

Task redundancy has been addressed since the early 1980s in the context of robotic manipulators under inverse kinematic control (Hanafusa, Yoshikawa, and Nakamura 1981; Nakamura, Hanafusa, and Yoshikawa 1987). For instance, a 7 DOF manipulator will be redundant with respect to a 5-dimensional task (e.g. controlling the end-effector's position and two orientation coordinates). In the previous example the manipulator would have 2 DOFs of redundancy. In the case of a humanoid the redundant space is an order of magnitude larger! A humanoid with 30 DOFs would have 25 DOFs of redundancy for the previous task. One of our goals in this chapter will be to characterize the redundant space with respect to operational tasks. This redundancy will be later exploited to simultaneously control manipulation, locomotion, balance, and postural tasks.

This chapter is organized as follows. In Section 2.1 we will discuss the representation of multi-legged robots at the kinematic and dynamic levels. In Section 2.2 we will develop an operational space controller and characterize the residual task redundancy.

2.1 Representation of Multi-Legged Robots

A fundamental characteristic of humanoid robots is their freedom to move anywhere in space under the supporting ground. Legged locomotion (see Figure 2.1) allows robots to virtually reach any place that humans can reach. In this section we will study the kinematics and dynamics of multi-legged robots under ground supports.

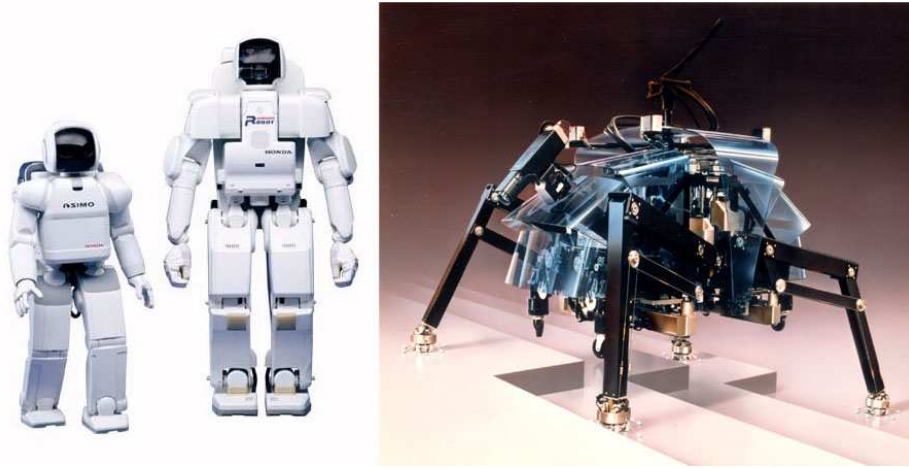


Figure 2.1: **Multi-legged robots:** The humanoid robots on the left side are Asimo and P3 by Honda Co. The robot on the right side is Titan IV developed by the Tokyo Institute of Technology (Hirose’s lab).

2.1.1 Kinematic and Dynamic Models

To create kinematic and dynamic representations under supporting constraints we will represent here multi-legged robots as free floating systems in contact with the ground and analyze the impact of the associated constraints and the resulting reaction forces.

Non-Fixed Branching Kinematic Model

To describe the movement of the robot in space we designate one of its body structures as the root base which can be arbitrarily displaced with respect to an inertial frame of reference. We describe the movement of the base according to its position and orientation in cartesian space. The root base is used to describe robot kinematic and dynamic quantities by exploiting the overall branching structure. In Figure 2.2 we show a depiction of a kinematic representation of a humanoid. \mathcal{R}_0 represents an inertial frame or reference (the global frame), and \mathcal{R}_b represents the frame or reference at the root base, in our case the hip

structure. The position and orientation of the robot in space is measured as the relative position and orientation of the root base with respect to the inertial frame, i.e.

$$x_b = \begin{pmatrix} x_{b,p} \\ x_{b,r} \end{pmatrix}. \tag{2.1}$$

Here $x_{b,p}$ and $x_{b,r}$ represent respectively the position and orientation of the base with respect to global coordinates. The position is normally represented in Cartesian coordinates, i.e. $x_{b,p} = (x, y, z)^T$, while the orientation is normally represented using Euler quaternion parameters, i.e. $x_{b,r} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$, or Euler angles, i.e. $x_{b,r} = (\psi, \theta, \phi)^T$. The main advantage of using Euler parameter representations is that they do not suffer from representation singularities. For a discussion on coordinate representations see (Khatib 2004). The spatial velocity of the base can be expressed in terms of linear and angular velocities, i.e.

$$\vartheta_b = \begin{pmatrix} v_b \\ \omega_b \end{pmatrix} \in \mathcal{R}^6, \tag{2.2}$$

where v_b and ω_b correspond to linear and angular velocities respectively.

In (Chang and Khatib 2000) a representation of ground-fixed branching mechanisms (i.e. robots with multiple limbs) and a collection of efficient algorithms for the computation of kinematic and dynamic quantities were presented. As mentioned above, in humanoids we need to use a non-fixed branching representation. However, the same fast algorithms developed by Chang can be applied to non-fixed systems by creating branching representations that include additional passive spherical and linear DOFs attached to the robot’s base representing the 6 DOFs of free motion in space (see Figure 2.3).

The humanoid system is therefore treated as a holonomic system with n actuated joints and 6 passive DOFs describing the position and orientation of its base.

Definition 2.1.1 (Robot generalized coordinates). *The robot position and orientation in space and the position of its joints can be fully described by the set*

$$\{x_b, q\} = \{x_{b,p}, x_{b,r}, q_1, q_2, \dots, q_n\}, \tag{2.3}$$

where the vector x_b represents the coordinates of the base link and the $n \times 1$ vector q represents actuated joint positions.

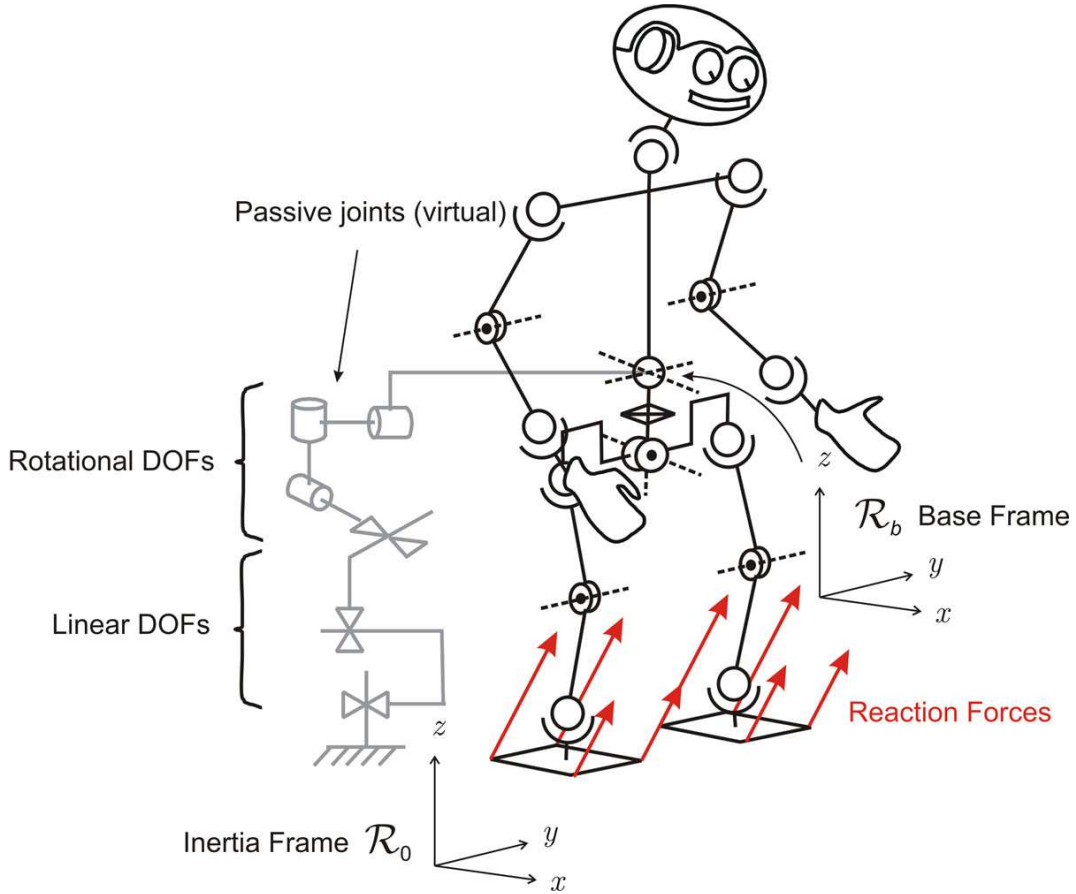


Figure 2.2: **Kinematic representation of a legged robot:** The free moving base is represented as a virtual spherical joint in series with three prismatic virtual joints. Reaction forces appear at the contact points due to gravity forces pushing the body against the ground.

Dynamic Model

Reaction forces appear on the supporting surfaces due to gravity forces and center of gravity (COG) accelerations. These reaction forces or contact constraints provide the means for stability, locomotion, and postural stance.

Using Lagrangian formalism and expressing the system's kinetic energy in terms of the individual link kinetic and potential energies we can derive the following equation of motion describing robot dynamics under supporting contacts

$$A \begin{pmatrix} \dot{v}_b \\ \ddot{q} \end{pmatrix} + b + g + J_s^T F_r = \begin{pmatrix} 0 \\ \Gamma \end{pmatrix}, \quad (2.4)$$

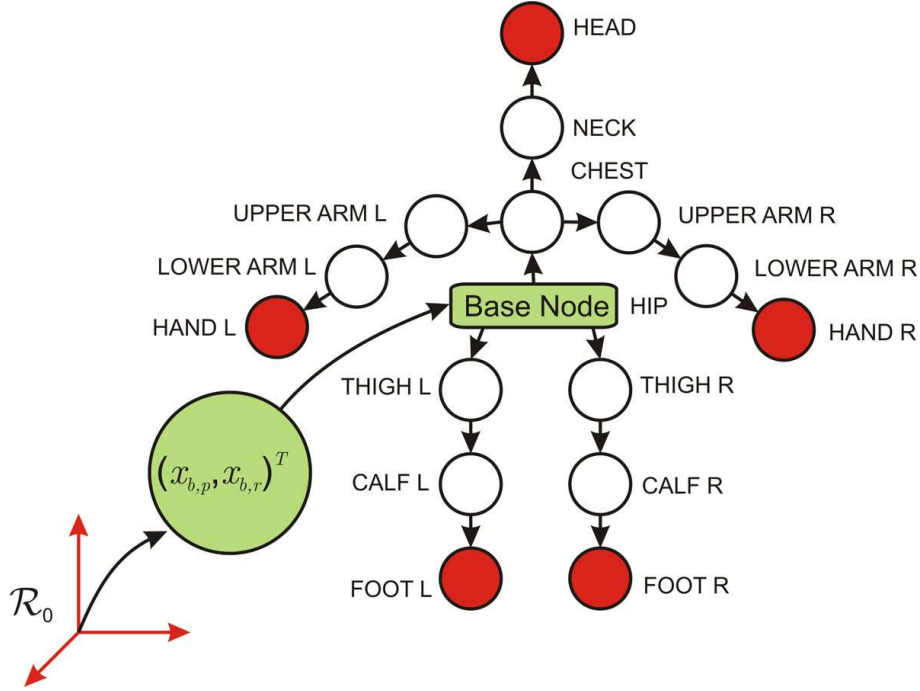


Figure 2.3: **Non-fixed branching representation:** The white and red nodes represent actuated links. The base node is unactuated with 6 passive DOFs describing its position and orientation with respect to the inertial frame of reference.

where A is the $(6+n) \times (6+n)$ inertia matrix, b and g are $(6+n) \times 1$ vectors of Coriolis/centrifugal and gravity forces respectively, and Γ is the $n \times 1$ vector of actuation torques. Notice that the actuation vector in the right hand side (RHS) of the previous equation has six zeros corresponding to the six passive DOFs associated with the robot's unactuated base. Moreover, the term $J_s^T F_r$ corresponds to the projection of reaction forces acting on the feet into forces acting in the passive and actuated DOFs and J_s corresponds to the Jacobian associated with all supporting links. The joint coordinate vector of all supporting links is

$$x_s \triangleq \begin{pmatrix} x_{s(rf)} \\ x_{s(lf)} \end{pmatrix}, \quad x_{s(i)} \triangleq \begin{pmatrix} x_{s(i),p} \\ x_{s(i),r} \end{pmatrix}, \quad (2.5)$$

where the subscripts rf and lf stand for right and left foot respectively, $x_{s(i)}$ is the position and orientation vector of the center at mass of the i -th supporting link, and $x_{s(i),p}$ and $x_{s(i),r}$ are position and orientation representations of the i -th supporting link respectively. The spatial velocity (Featherstone 1987) at the support points can therefore be expressed

as

$$\vartheta_s \triangleq \begin{pmatrix} \vartheta_{s(rf)} \\ \vartheta_{s(lf)} \end{pmatrix} = J_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad \vartheta_{s(i)} \triangleq \begin{pmatrix} v_{s(i)} \\ \omega_{s(i)} \end{pmatrix} \in \mathcal{R}^{12}, \quad (2.6)$$

where $\vartheta_{s(i)}$ corresponds to the spatial velocity of the i -th supporting link, $v_{s(i)}$ and $\omega_{s(i)}$ are linear and angular velocities of the i -th supporting link respectively, and J_s is the basic Jacobian (Khatib 2004) associated with all supporting links.

F_r corresponds to the sum of reaction forces and moments projected into link center of masses (see Figure 2.4), i.e.

$$F_r \triangleq \begin{pmatrix} F_{r(rf)} \\ F_{r(lf)} \end{pmatrix} \in \mathcal{R}^{12}. \quad (2.7)$$

We can further express the above reaction forces in terms of linear forces and moments as

$$F_{r(i)} \triangleq \begin{pmatrix} f_{r(i)} \\ m_{r(i)} \end{pmatrix} \in \mathcal{R}^6, \quad (2.8)$$

where $f_{r(i)}$ is the vector of linear forces and $m_{r(i)}$ is the vector of moments of the i -th supporting link. Each of these components can be further expressed as

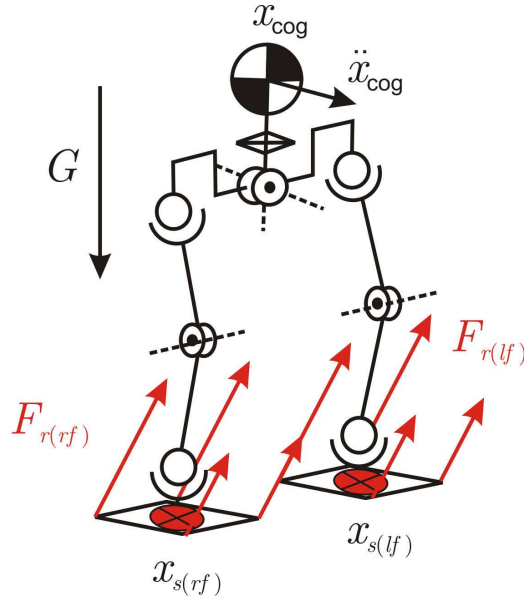


Figure 2.4: **Reaction forces on the feet:** Linear reaction forces have components due to the action of the gravity field G and due to vertical COG accelerations, and tangential components due to horizontal COG accelerations. These forces create reaction moments as well between the gravity centers on the feet and the center of pressure points.

$$\mathbf{f}_{r(i)} = \mathbf{f}_{\text{friction}(i)} + \mathbf{f}_{\text{gravity}(i)}, \quad (2.9)$$

$$\mathbf{m}_{r(i)} = \mathbf{f}_{r(i)} \times (\mathbf{x}_{s(i)} - \mathbf{x}_{\text{cop}(i)}). \quad (2.10)$$

Here the vector of linear forces has been obtained by adding frictional and gravity terms and the vector of moments has been obtained by calculating the cross product between linear forces projected in the link's COM and the distance between the link's COM and its center of pressure (COP).

Reaction forces on the feet translate into forces acting on the robot's passive and actuated DOFs and can be expressed according to the principle of virtual work as

$$\Gamma_s \triangleq J_s^T F_r \in \mathcal{R}^{(6+n)}, \quad (2.11)$$

where Γ_s is the vector of forces acting on both passive and actuated DOFs.

The following expression reveals the contribution from the passive chain (the virtual joints describing the movement of the base) and the actuated joints on the legs (see Figure 2.6),

$$J_s = \begin{pmatrix} V_{b,s(rf)} & J_{s(rl)} & 0 & 0_{(\text{upper})} \\ V_{b,s(lf)} & 0 & J_{s(ll)} & 0_{(\text{upper})} \end{pmatrix} \in R^{12 \times (6+n)}, \quad (2.12)$$

where

$$V_{b,s(i)} \triangleq \begin{pmatrix} I & \hat{p}_{b,s(i)} \\ 0 & I \end{pmatrix} \quad (2.13)$$

is a transformation matrix which maps angular movement of the base to linear velocities at the i -th support point (see discussion on macro/mini structures in Khatib 2004), $p_{b,s(i)}$ corresponds to the distance vector between the i -th support link and the robot's base, $\hat{p}_{b,s(i)}$ is the cross product operator associated with the position vector, $J_{s(rl)}$ and $J_{s(ll)}$ are block matrices corresponding to displacements of joints on the right and left legs respectively with respect to the robot's base (see Figure 2.6), and the term $0_{(\text{upper})}$ represents the null contribution of upper body motions to movement on supporting points. Therefore, the above support Jacobian can be decomposed into passive and actuated components corresponding to base and robot DOFs, i.e.

$$J_s = \begin{pmatrix} J_{sb} & J_{sr} \end{pmatrix} \quad (2.14)$$

The generalized inertia matrix shown in (2.4) can also be expressed in terms of passive

and actuated components as follows

$$A = \begin{pmatrix} A_{bb} & A_{br} \\ A_{br}^T & A_{rr} \end{pmatrix} \in \mathcal{R}^{(6+n) \times (6+n)}, \quad (2.15)$$

where A_{bb} corresponds to the inertia felt at the base, A_{rr} corresponds to the joint space inertia of the robot, and A_{br} corresponds to the inertial weight between accelerations of the base and the resulting torques on the actuated joints. Notice that A_{bb} is independent on the position and orientation of the robot in space, as there is no physical linkage connecting the robot's base to the inertial frame of reference. In fact, the entire inertia matrix A is independent of the global position and orientation of the robot in space.

2.1.2 Impact of Supporting Contacts

Supporting contacts at the feet, and in general in any other place in the robot's body provide the support to realize advanced behaviors. Therefore, they affect the robot's motion at the kinematic, dynamic, and control levels. Supporting contacts should be distinguished from manipulation interactions because their role is indirect, i.e. to provide stability above the ground.

As shown in Equation (2.4), reaction forces against the ground translate into forces acting on passive and actuated DOFs. With the premise that stable balance is maintained and that internal forces are controlled to keep the feet flat against the ground, no relative movement occurs between contact points and the supporting ground. Therefore, relative velocities and accelerations at the contact points are equal to zero. The following set of non-holonomic constraints express this condition (Yamane and Nakamura 2003)

$$\vartheta_s = \begin{pmatrix} \vartheta_{s(rf)} \\ \vartheta_{s(lf)} \end{pmatrix} = 0, \quad \dot{\vartheta}_s = \begin{pmatrix} \dot{\vartheta}_{s(rf)} \\ \dot{\vartheta}_{s(lf)} \end{pmatrix} = 0. \quad (2.16)$$

We first analyze the impact of the above constraints on the robot's equation of motion. Let us first derive the equation of motion at the supporting links. By right-multiplying (2.4) by the term $J_s A^{-1}$ and considering the equality

$$\dot{\vartheta}_s = J_s \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} + \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (2.17)$$

we obtain the following equation of motion for the supporting links,

$$\dot{\vartheta}_s - \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + J_s A^{-1}(b + g) + J_s A^{-1} J_s^T F_r = J_s A^{-1} S^T \Gamma. \quad (2.18)$$

Definition 2.1.2 (Actuation matrix). *The matrix*

$$S \triangleq \begin{pmatrix} 0_{n \times 6} & I_{n \times n} \end{pmatrix} \quad (2.19)$$

is an actuation matrix selecting actuated DOFs and is used to express the RHS of (2.4) in a compact form, i.e.

$$\begin{pmatrix} 0_{6 \times 1} \\ \Gamma \end{pmatrix} = S^T \Gamma. \quad (2.20)$$

Solving (2.18) for the constraint $\dot{\vartheta}_s = 0$ we obtain an estimate of the reaction forces in terms of the actuation torques, i.e.

$$F_r = \bar{J}_s^T S^T \Gamma - \bar{J}_s^T (b + g) + \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}. \quad (2.21)$$

where

$$\Lambda_s \triangleq (J_s A^{-1} J_s^T)^{-1} \quad (2.22)$$

is the apparent inertia at the supporting links, and

$$\bar{J}_s \triangleq A^{-1} J_s^T \Lambda_s \quad (2.23)$$

is the dynamically consistent generalized inverse of J_s . Using the above equation in (2.4) we obtain the following constrained equation of motion

$$A \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} + N_s^T (b + g) + J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = N_s^T S^T \Gamma. \quad (2.24)$$

Definition 2.1.3 (Dynamically consistent null space of supporting contacts). *The matrix*

$$N_s \triangleq I - \bar{J}_s J_s, \quad (2.25)$$

is referred to as the dynamically consistent null-space matrix (Khatib 1987) of J_s and defines a generalized space of motion with no acceleration or force coupling effects on the supporting links.

Let us consider the following properties of N_s :

Property 2.1.1 (Idempotence of N_s). *The following equality holds*

$$(N_s)^2 = N_s. \tag{2.26}$$

Proof. The above equality can be proven by using (2.25) and the general property of generalized inverses $J_s \bar{J}_s J_s = J_s$. \square

Property 2.1.2 (Commutation of N_s with respect to A^{-1}). *The following equalities hold*

$$N_s A^{-1} = A^{-1} N_s^T = N_s A^{-1} N_s^T. \tag{2.27}$$

Proof. It is easy to prove that $N_s A^{-1} = A^{-1} N_s^T$ by using (2.25) and (2.23). Then the following equality holds, $N_s A^{-1} N_s^T = (N_s)^2 A^{-1}$ which using (2.26) becomes equal to $N_s A^{-1}$. \square

2.2 Operational Space Control

To realize complex behaviors, a humanoid needs to simultaneously control multiple task points. For example to create the walking behavior shown in Figure 2.5 the robot needs to control the position or acceleration of the COG, the position and orientation of the swinging foot (in the case of single support stance), and the orientation of the head. Other tasks such as hand manipulation could also be simultaneously controlled. The residual DOFs are used to control the robot’s posture with the methods we will describe in Chapter 4.

To characterize the overall behavior, we consider the vector of task points (see Figure 2.5)

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}, \tag{2.28}$$

where each x_i describes the position and orientation of the i -th task point and N is the number of task points.

To execute a desired movement, each task point is controlled to accomplish a specific

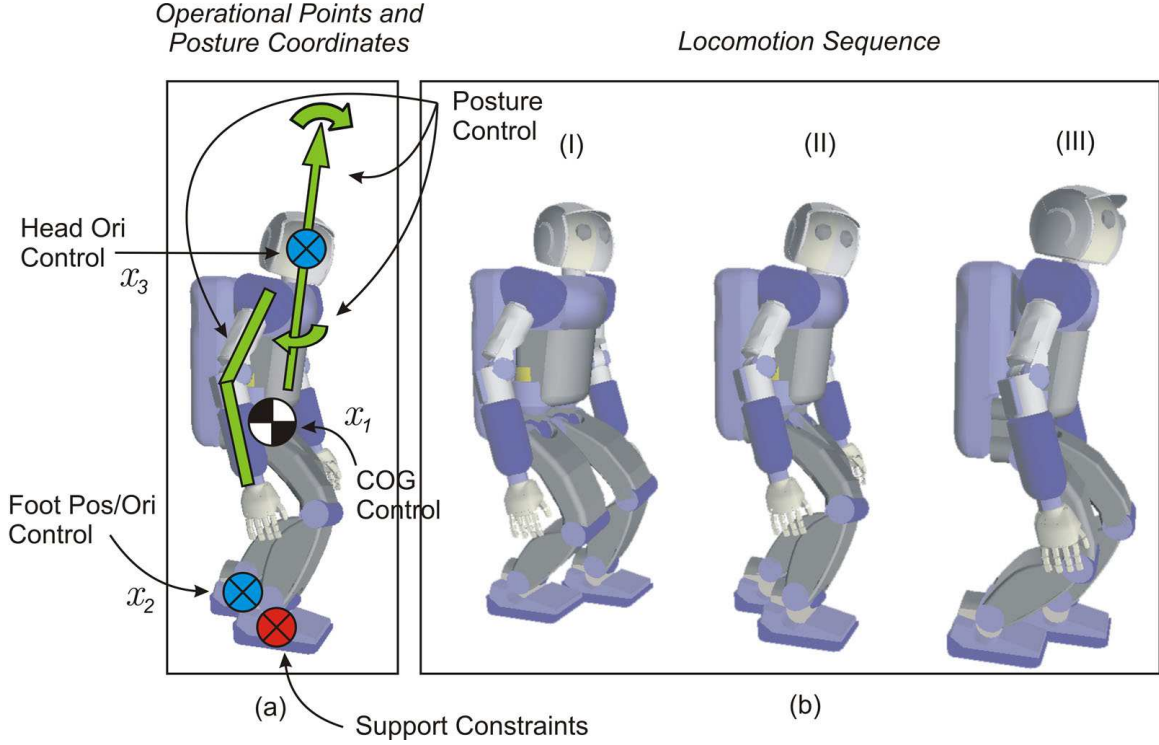


Figure 2.5: **Whole-body walking behavior:** Figure (b) depicts a walking sequence from an actual experiment. Figure (a) shows the task points, x_i , that need to be actively controlled to achieve the desired behavior. This includes COG control (shown with a black and white symbol), position and orientation of the swinging foot (shown with a blue cross), and orientation of the head (also shown with a blue cross). The stable foot acts as a support constraint. Posture DOFs are shown with green lines and arrows.

goal g_i . The aggregation of goals can be characterized by the following vector

$$g = \begin{pmatrix} g_1 \\ g_2 \\ \dots \\ g_N \end{pmatrix}. \quad (2.29)$$

These goals correspond to desired position, force, or acceleration vectors, i.e.

$$g_i = \begin{cases} x_{\text{des}(i)}, & \text{if position task} \\ f_{\text{des}(i)}, & \text{if force task} \\ (x_{\text{des}(i)}, f_{\text{des}(i)}), & \text{if hybrid task} \\ a_{\text{des}(i)}, & \text{if acceleration task} \end{cases} \quad (2.30)$$

In this chapter, we will consider first the control of simple behaviors, such as maintaining balance stability. Static or dynamic balance can be achieved by controlling the robot's horizontal COG components. On a humanoid equipped with 30 DOFs, the control of the horizontal COG will leave 28 degrees of additional movement redundancy. This redundancy will be mathematically characterized here and later exploited to control the robot's posture. In the next chapter we will consider the control of complex behaviors involving the simultaneous coordination of multiple low-level tasks.

2.2.1 Task Kinematics Under Supporting Constraints

When a single or multiple feet or hands are in contact with supporting surfaces, the robot's motion is constrained by the acting reaction forces. In the previous section we characterized whole-body kinematic and dynamic representations under supporting constraints. Based on these representations, it is the objective of this section to characterize and control operational tasks under supporting constraints.

An arbitrary task point can be represented by its position and orientation with respect to the global frame of reference, i.e.

$$x = \begin{pmatrix} x_p \\ x_r \end{pmatrix}, \quad (2.31)$$

where x_p is a position representation and x_r is an orientation representation. Position and orientation representations can vary depending on the type of task to be implemented as explained in (Khatib 2004). Moreover, a subset of these coordinates can be considered for tasks involving fewer than 6 DOFs such as in reaching or looking tasks.

Task velocities can be expressed using arbitrary representations by considering linear and angular velocities of the task point and transforming them to the desired representation, i.e.

$$\dot{x} = E(x) \begin{pmatrix} v \\ \omega \end{pmatrix}, \quad (2.32)$$

where $E(x)$ is a representation transformation matrix and is described in (Khatib 2004) and v and ω are linear and angular velocities of the task point respectively. The instantaneous kinematics of arbitrary task points is expressed in terms of base and joint velocities as

$$\dot{x} = J \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad J = E(x)J_0, \quad (2.33)$$

where J is the task Jacobian in task coordinates and can be derived from the basic task Jacobian J_0 as explained in (Khatib 2004). When multiple task points are aggregated, i.e.

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}, \quad x_i = \begin{pmatrix} x_{i,p} \\ x_{i,r} \end{pmatrix}, \quad (2.34)$$

where $x_{i,p}$ and $x_{i,r}$ correspond to the position and orientation of task points (or a subset of these coordinates), the combined Jacobian matrix is equal to

$$J = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_N \end{pmatrix}. \quad (2.35)$$

Constrained Kinematics

Because the robot is constrained by the supporting ground, the position of its base can be derived from the position of actuated joints alone. In turn, the position of arbitrary task points can be obtained from joint positions only. Let us study this dependency more closely. The velocity constraint on the supporting feet, i.e. $\vartheta_s = 0$ (2.16), means that base and joint velocities are not arbitrary. They are quantities that lie in the null space of the support Jacobian, leading to the following expression

$$\begin{pmatrix} \vartheta_b^* \\ \dot{q}^* \end{pmatrix} \triangleq N_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (2.36)$$

where ϑ_b and \dot{q} are arbitrary vectors of base and joint velocities, ϑ_b^* and \dot{q}^* are the corresponding constrained quantities, and N_s is the dynamically consistent null-space (Khatib 1987) of J_s as shown in (2.25). Moreover, constrained joint velocities alone can be obtained by multiplying the above equation by the actuation matrix S , i.e.

$$\dot{q}^* = SN_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}. \quad (2.37)$$

When the robot is in single support stance, the matrix SN_s is full rank and therefore \dot{q}^*

can take arbitrary values. However, when the robot is in double support stance or multi-limb stance SN_s is not full rank and as a result \dot{q}^* cannot take arbitrary values. This is due to the presence of close loops imposed by the limbs in contact. Solving the above equation, any vector of base and joint velocities can be decomposed into a component that depends on joint velocities only and a component that lies in the residual null space of motion, i.e.

$$\begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = \overline{SN}_s \dot{q}^* + (I - \overline{SN}_s SN_s) \begin{pmatrix} \vartheta_{b,0} \\ \dot{q}_0 \end{pmatrix}, \tag{2.38}$$

where \overline{SN}_s is a support consistent generalized inverse of SN_s that will be derived in a few lines, $I - \overline{SN}_s SN_s$ is a null space basis associated with the matrix SN_s , and $\vartheta_{b,0}$ and \dot{q}_0 are arbitrary vectors of base and joint velocities operating in the null space of SN_s .

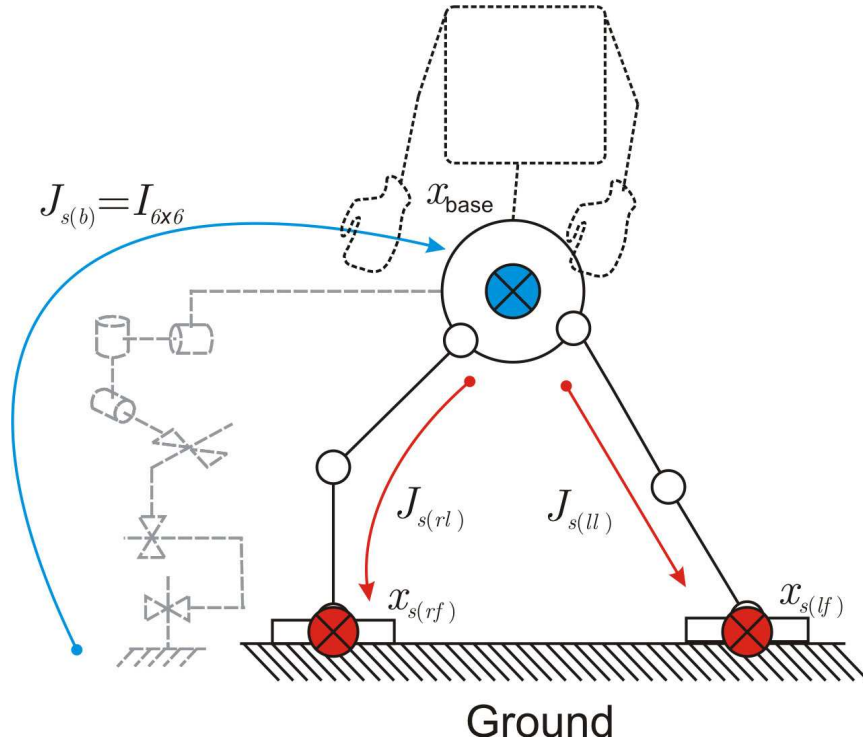


Figure 2.6: **Dependency between base and joint displacements:** The Jacobian $J_{s(b)}$ — normally equal to the identity matrix — corresponds to displacements of the robot’s base, while $J_{s(rl)}$ and $J_{s(ll)}$ correspond to displacements of the right and left legs respectively. Because of support constraint on the feet, base displacements can be represented as a function of joint displacements on the right or left legs.

Using (2.38) in (2.36) leads to the equality

$$\begin{pmatrix} \vartheta_b^* \\ \dot{q}^* \end{pmatrix} = \overline{SN_s} \dot{q}^*, \quad (2.39)$$

where we have used the equalities $N_s \overline{SN_s} = \overline{SN_s}$ that will be shown in (2.43) and $N_s(I - \overline{SN_s} SN_s) = 0$ which can be derived using (2.43) and (2.44). In Figure 2.6 we depict a constrained scenario illustrating the dependency between base displacements and joint displacements justifying the above expression.

Lemma 2.2.1 (Support consistent generalized inverse of SN_s). *A generalized inverse of SN_s that fulfills the velocity constraint $\vartheta_s = 0$ is the dynamically weighted generalized inverse of SN_s with weight equal to A^{-1} , i.e.*

$$\overline{SN_s} \triangleq A^{-1} (SN_s)^T (SN_s A^{-1} (SN_s)^T)^+. \quad (2.40)$$

(Remark: In single support phase, SN_s is full rank and therefore the above pseudo-inverse becomes an inverse.)

Proof. To show that the above expression is consistent with supporting constraints we use it in (2.39) and apply the resulting expression in (2.6), yielding the desired cancelation of terms, i.e.

$$\vartheta_s = J_s \begin{pmatrix} \vartheta_b^* \\ \dot{q}^* \end{pmatrix} = J_s A^{-1} N_s^T S^T (SN_s A^{-1} (SN_s)^T)^+ \dot{q}^* = 0, \quad (2.41)$$

where we have used the equality $J_s A^{-1} N_s^T = 0$. This last equality can be demonstrated using the expression of N_s given in (2.25). \square

Definition 2.2.1 (Constrained projection of A^{-1}). *The following expression appearing in (2.40) is a constrained projection of the robot's inverse inertia matrix*

$$\Phi^* \triangleq SN_s A^{-1} (SN_s)^T. \quad (2.42)$$

The term $\overline{SN_s}$ will reappear when formulating operational space controllers. Two interesting properties are associated with this term.

Property 2.2.1 (Invariance of $\overline{SN_s}$ with respect to N_s). *The following equality holds*

$$N_s (\overline{SN_s}) = \overline{SN_s}. \quad (2.43)$$

Proof. This equality can be demonstrated by using the expression of N_s given in (2.25) and the expression of $\overline{SN_s}$ given in (2.40). \square

Property 2.2.2 (Correspondence between SN_s and N_s). *The following equality holds*

$$\overline{SN_s} SN_s = N_s. \quad (2.44)$$

Proof. This equality can be demonstrated by using (2.37) in (2.39), which leads to the equality

$$\begin{pmatrix} \vartheta_b^* \\ \dot{q}^* \end{pmatrix} = \overline{SN_s} SN_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (2.45)$$

where ϑ_b and \dot{q} are arbitrary velocities. Comparing the above expression with (2.36) leads to the desired equality. \square

When using (2.39), task velocities can be expressed in terms of joint velocities alone, i.e.

$$\dot{x} = J \begin{pmatrix} \vartheta_b^* \\ \dot{q}^* \end{pmatrix} = J \overline{SN_s} \dot{q}^*. \quad (2.46)$$

Definition 2.2.2 (Support consistent reduced Jacobian). *The term $J\overline{SN_s}$ in the above equation, acts as a constrained Jacobian mapping joint velocities into task velocities and we will refer to it using the symbol*

$$J^* \triangleq J\overline{SN_s}. \quad (2.47)$$

This expression is motivated by the dependency of base velocities on joint velocities, as shown in Figure 2.6.

Definition 2.2.3 (Support consistent full Jacobian). *The following constrained expression determines the mapping between arbitrary base and joint velocities to task velocities*

$$J_{t|s} \triangleq JN_s. \quad (2.48)$$

Here we use the subscript $t|s$ to indicate that the task Jacobian is projected in the space consistent with supporting constraints. The above Jacobian appears when we apply the

constrained velocity vector (2.36) into (2.33), yielding the equality

$$\dot{x} = J \begin{pmatrix} \vartheta_b^* \\ \dot{q}^* \end{pmatrix} = JN_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (2.49)$$

indicating that $J_{t|s}$ projects arbitrary base and joint velocities into quantities that are consistent with supporting constraints.

Property 2.2.3 (Correspondence between $J_{t|s}$ and J^*). *The following equalities hold,*

$$J_{t|s} = J^*(SN_s), \quad (2.50)$$

$$J^* = J_{t|s}(\overline{SN_s}). \quad (2.51)$$

Proof. The above equalities can be demonstrated using (2.43) and (2.44). □

Kinematic Singularities

Using (2.47) we can study the kinematic singularities of a given task representation under supporting constraints. In fact, the constrained Jacobian J^* reflects both structural and contact singularities whereas the full Jacobian J reflects only structural singularities.

In Figure 2.7 we study an example on kinematic singularities. A task associated with the COG's cartesian position is controlled using operational space methods that will be described later in this chapter. We assume balance is properly maintained and the robot's feet are stable against the ground. The COG's position and its Jacobian can be expressed as

$$x = \frac{1}{M} \sum_{i=1}^n m_i x_{\text{cog}(i)} \in \mathcal{R}^3, \quad J = \frac{1}{M} \sum_{i=1}^n m_i J_{\text{cog}(i)} \in \mathcal{R}^{3 \times (6+n)}, \quad (2.52)$$

where M is the robot's total mass, m_i and $x_{\text{cog}(i)}$ are the mass and COG position of i -th link, and $J_{\text{cog}(i)}$ is the Jacobian of the same link. To study task singularities we consider the condition number of the constrained Jacobian (2.47) of the COG

$$\kappa(J_{\text{cog}}^*) \triangleq \frac{\sigma_1(J_{\text{cog}}^*)}{\sigma_3(J_{\text{cog}}^*)}, \quad J_{\text{cog}}^* \triangleq J_{\text{cog}} \overline{SN_s}, \quad (2.53)$$

where $\sigma_i(J_{\text{cog}}^*)$ is the i -th singular value of the constrained Jacobian, as well as the condition number of the full Jacobian $\kappa(J_{\text{cog}})$. Data on both quantities is shown in Figure 2.7. When the legs reach full stretch, the COG's vertical position cannot move further upwards. As a result, the task's constrained Jacobian becomes singular and its condition number grows

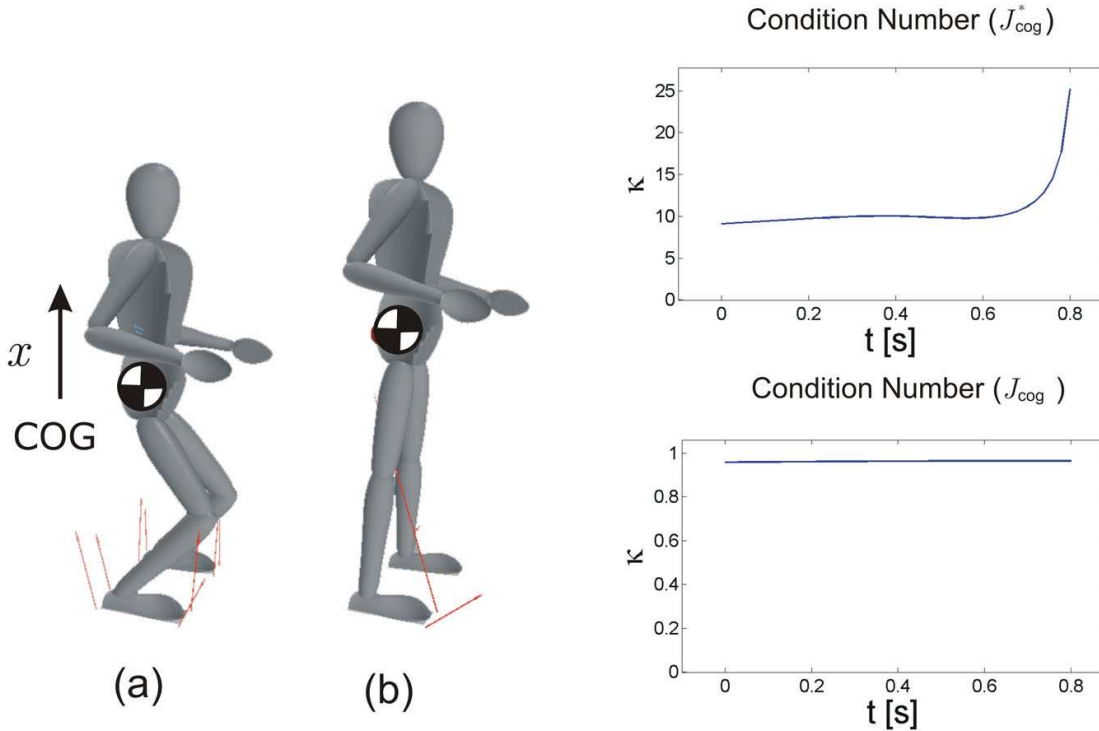


Figure 2.7: **Kinematic singularities under support constraints:** This image corresponds to an example where the robot's COG is controlled to move from position (a) to position (b). When moving towards position (b) the robot's legs reach maximum stretch. As a result the COG's constrained Jacobian becomes singular. In contrast, the condition number of the COG's full Jacobian stays invariant because it does not account for contact constraints.

towards infinity. In contrast, the full Jacobian does not lose rank because it does not reflect the singularities due to contact constraints on the feet.

2.2.2 Task Dynamics and Control

We consider here the control of a single task point. For instance, in Figure 2.8 we depict a robot performing a contact task involving inserting screws in a wooden structure. Reaction forces on the robot's feet and on the tip of the screwgun appear due to gravity effects, COG accelerations, and task contacts. While feet reaction forces are used for support and balance, task forces are carefully controlled to insert screws.

The robot's dynamic equation now includes an additional term involving reaction forces

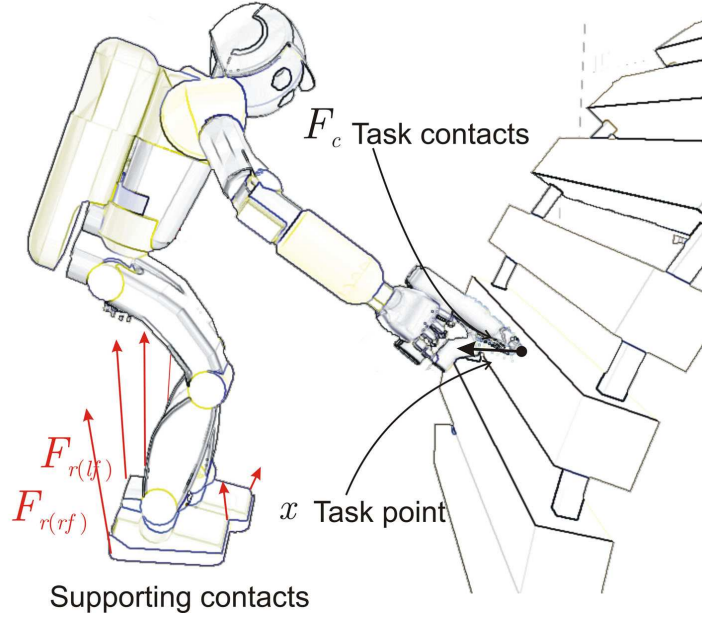


Figure 2.8: **Contact forces during task interactions:** This image depicts reaction forces appearing at the bottom of the robot’s feet and on the tip of the screwgun during a contact task. While tool forces are carefully controlled to insert screws, feet reaction forces are used for support and balance.

at the tip of the tool, i.e.

$$A \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} + b + g + J^T F_c + J_s^T F_r = S^T \Gamma, \quad (2.54)$$

where J is the Jacobian at the task point (in this case the tip of the screwgun), F_c is a 6×1 vector of reaction forces acting on the task point, J_s is the joint Jacobian associated with the supporting links, and F_r is the sum of reaction forces on these links. Similarly to (2.21), we solve the above equation of motion for F_r using the support constraint $\dot{\vartheta}_s = 0$, yielding the equality

$$F_r = \bar{J}_s^T S^T \Gamma - \bar{J}_s^T J^T F_c - \bar{J}_s^T (b + g) + \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}. \quad (2.55)$$

Notice that the reaction forces on the feet depend not only on control torques but also on contact forces acting at the tip of the tool.

Using the above equality in (2.54) yields the equation

$$A \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} + N_s^T(b + g) + J_{t|s}^T F_c - J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = (SN_s)^T \Gamma, \quad (2.56)$$

where we $J_{t|s}$ is the constrained Jacobian presented in (2.48). Notice that in the above equation the reaction forces on the tool's tip are projected by the constrained Jacobian $J_{t|s}$, due to feet stability constraints.

Definition 2.2.4 (Dynamically consistent generalized inverse of $J_{t|s}$). *The following expression is referred to as the dynamically consistent generalized inverse of $J_{t|s}$*

$$\bar{J}_{t|s} \triangleq A^{-1} J_{t|s}^T \Lambda_{t|s}, \quad (2.57)$$

where

$$\Lambda_{t|s} \triangleq (J_{t|s} A^{-1} J_{t|s}^T)^{-1}, \quad (2.58)$$

is the task space inertia matrix under supporting constraints. Because $\bar{J}_{t|s}$ is used to obtain the equation of motion in task space as we will show in (2.59) and defines the dynamically consistent null space of motion shown in (2.77) we will refer to it as the dynamically consistent generalized inverse (Khatib 1987) of $J_{t|s}$.

The tasks' equation of motion can be obtained by left multiplying (2.56) by the transpose of the dynamically consistent generalized inverse of the constrained Jacobian, $\bar{J}_{t|s}^T$ yielding the following task space equation of motion

$$\Lambda_{t|s} \ddot{x} + \mu_{t|s} + p_{t|s} + F_c = \bar{J}_{t|s}^T (SN_s)^T \Gamma, \quad (2.59)$$

where $\mu_{t|s}$ and $p_{t|s}$ are Coriolis/centrifugal and gravity terms with the following expressions

$$\mu_{t|s} \triangleq \bar{J}_{t|s}^T b - \left(\Lambda_{t|s} \dot{J}_{t|s} + \bar{J}_{t|s}^T J_s^T \Lambda_s \dot{J}_s \right) \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (2.60)$$

$$p_{t|s} \triangleq \bar{J}_{t|s}^T g, \quad (2.61)$$

where we have used (2.49) and its derivative. Moreover, to obtain (2.59) we have used the following property:

Property 2.2.4 (Invariance of $\bar{J}_{t|s}$ with respect to N_s). *The following equality holds*

$$N_s \bar{J}_{t|s} = \bar{J}_{t|s}. \quad (2.62)$$

Proof. This equality can be demonstrated by using the expression of $\bar{J}_{t|s}$ given in (2.57), the equality $N_s A^{-1} = A^{-1} N_s^T$ shown in (2.27), and the property $(N_s)^2 = N_s$ shown in (2.26). \square

Theorem 2.2.1 (Operational space control). *The following torque vector yields linear control of task forces and accelerations*

$$\Gamma = (\overline{SN_s})^T J_{t|s}^T F, \quad (2.63)$$

or equivalently

$$\Gamma = J^{*T} F, \quad (2.64)$$

where we have used the equality $J^* = J_{t|s}(\overline{SN_s})$ shown in (2.51).

Proof. In the above equations, F is a vector of control forces in task space. Applying either of the above expressions to (2.59) and using the equality $\overline{SN_s} SN_s = N_s$ shown in (2.44) and the equality $N_s \bar{J}_{t|s} = \bar{J}_{t|s}$ shown in (2.62) leads to the following task space equation of motion

$$\Lambda_{t|s} \ddot{x} + \mu_{t|s} + p_{t|s} + F_c = F, \quad (2.65)$$

which defines a linear relationship between control forces, contact forces, and task accelerations. \square

Choosing F appropriately we can implement a variety of control strategies such as hybrid position/force control or acceleration control.

Corollary 2.2.1 (Motion control). *In the absence of contact interactions (i.e. $F_c = 0$), the following control vector will yield linear control of task accelerations*

$$F = \Lambda_{t|s} a^{ref} + \mu_{t|s} + p_{t|s}, \quad (2.66)$$

where a^{ref} is a reference acceleration vector.

Proof. It is straightforward to verify that applying the above control force to (2.65) will yield the linear behavior

$$\ddot{x} = a^{ref}. \quad (2.67)$$

□

Choosing appropriately a^{ref} we can implement a variety of motion control strategies. Alternatively, we can implement hybrid position force control strategies by choosing the control force

$$F = \Lambda_{t|s} \Omega_m a^{ref} + \Omega_f F^{ref} + \mu_{t|s} + p_{t|s} \quad (2.68)$$

where the projection matrices Ω_m and Ω_f are used to split the force vector into motion and force components (i.e. tangential or normal to the contacting surface), and a^{ref} and F^{ref} are control policies in acceleration and force space respectively. For more details on hybrid position/force control see (Khatib 1987) and (Featherstone, Thiebaut, and Khatib 1999).

Multi-Task Control

When multiple tasks are simultaneously controlled as part of a complex behavior, the joint tasks' equation of motion is equivalent to (2.59), however the quantities correspond to multiple task points as shown in (2.34) and (2.35). Therefore, the overall constrained Jacobian corresponds to the aggregation of individual constrained terms, i.e.

$$J_{t|s} \triangleq \begin{pmatrix} J_{1|s} \\ J_{2|s} \\ \vdots \\ J_{N|s} \end{pmatrix}, \quad J_{i|s} \triangleq J_i N_s, \quad (2.69)$$

where J_i are Jacobians of individual task points (see Figure 2.5).

Moreover, the joint task inertia matrix can be expressed in terms of blocks corresponding to the different task points, i.e.

$$\Lambda_{t|s} = (J_{t|s} A^{-1} J_{t|s}^T)^{-1} = \begin{pmatrix} \Lambda_{11|s} & \Lambda_{12|s} & \cdots & \Lambda_{1N|s} \\ \Lambda_{21|s} & \Lambda_{22|s} & \cdots & \Lambda_{2N|s} \\ \vdots & \vdots & & \vdots \\ \Lambda_{N1|s} & \Lambda_{N2|s} & \cdots & \Lambda_{NN|s} \end{pmatrix}, \quad (2.70)$$

where the diagonal terms $\Lambda_{ii|s}$ correspond to the inertia felt at the i -th task point when a force is applied to the same point and the off diagonal terms $\Lambda_{ij|s}$ correspond to the inertia felt at the j -th task point when a force is applied to the i -th task point.

The control vector is equivalent to Equations (2.63) and (2.66), however the reference acceleration and force vectors now include control policies for all operational points.

Corollary 2.2.2 (Multi-task motion control). *The following torque vector yields simultaneous linear control of multiple task accelerations*

$$\Gamma = (\overline{SN}_s)^T J_{t|s}^T \left(\Lambda_{t|s} a^{ref} + \mu_{t|s} + p_{t|s} \right), \quad (2.71)$$

or equivalently

$$\Gamma = J^{*T} \left(\Lambda_{t|s} a^{ref} + \mu_{t|s} + p_{t|s} \right), \quad (2.72)$$

where we have used the equality $J^* = J_{t|s} \overline{SN}_s$ shown in (2.51) and a^{ref} is equal to

$$a^{ref} \triangleq \begin{pmatrix} a_1^{ref} \\ a_2^{ref} \\ \vdots \\ a_N^{ref} \end{pmatrix}. \quad (2.73)$$

Here the terms a_i^{ref} correspond to individual control policies.

Proof. Once more, it is straightforward to verify that using the previous torque vector in (2.65) — now corresponding to the equation of motion of all tasks — yields the following set of linear equalities

$$\forall i, \ddot{x}_i = a_i^{ref}. \quad (2.74)$$

□

Although aggregating tasks into a macro vector is an attractive solution for whole-body control it entails several problems. First, in the presence of modeling errors, there will be a coupling effect between task motions. Second, during conflicting scenarios — for instance, when joint limits are reached — there could be potential constraint violations. These problems will be overcome in the next chapter when we will describe prioritized multi-task control.

2.2.3 Characterization of the Residual Task Redundancy

To complete the proposed operational space formulation for multi-legged robots we will characterize here the residual task redundancy. This redundancy will be used to control additional operational tasks as well as the robot’s postural stance. We characterize redundant torques by adding an additional term to (2.63) with null contribution to tasks motion and forces.

Theorem 2.2.2 (Whole-body control structure). *The following control structure provides linear control of task forces and accelerations and defines the task's residual movement redundancy*

$$\Gamma = (\overline{SN_s})^T J_{t|s}^T F + N^{*T} \Gamma_0. \quad (2.75)$$

Here N^* is the null space matrix of $J_{t|s}(\overline{SN_s})$ and will be shown in (2.77) and Γ_0 is an arbitrary torque vector. Equivalently, the above whole-body control torque can be written as

$$\Gamma = J^{*T} F + N^{*T} \Gamma_0, \quad (2.76)$$

where we have used the correspondence between $J_{t|s}$ and J^* shown in (2.51). Notice that N^* is also the null space associated with J^* as will be shown in (2.78).

Proof. While the term $J^{*T} F$ provides linear control of task forces and accelerations as discussed in Theorem 2.2.1, the above null space term provides no coupling effects between null space torques and task space forces or accelerations. In turn, the vector Γ_0 can be used to control the robot's postural motion or additional operational tasks without interfering with the primary task. \square

Corollary 2.2.3 (Dynamically consistent null space matrix). *The following null space matrix projects secondary control torques Γ_0 into null task forces and accelerations*

$$N^* \triangleq I - (SN_s) \overline{J}_{t|s} J_{t|s} (\overline{SN_s}). \quad (2.77)$$

An equivalent expression is

$$N^* \triangleq I - \overline{J}^* J^*, \quad (2.78)$$

where $J^* = J_{t|s}(\overline{SN_s})$ as shown in (2.51) and \overline{J}^* is the dynamically consistent generalized inverse of J^* as shown in (2.81) and can be shown to be equal to $(SN_s) \overline{J}_{t|s}$.

Proof.

1. To verify that the expression (2.77) is valid we first plug it into (2.75) and the resulting term is plugged into (2.59). We then need to demonstrate that the resulting null space term vanishes away. This can be demonstrated by using the equality $\overline{SN_s} SN_s = N_s$ (2.44) and the property $N_s \overline{J}_{t|s} = \overline{J}_{t|s}$ (2.62).
2. We also need to show that $\overline{J}^* = (SN_s) \overline{J}_{t|s}$. To do that we first consider the expression of \overline{J}^* given in (2.81). The following equalities can be derived using the expression of

Φ^* (2.42) and the correspondence between J^* and $J_{t|s}$ shown in (2.51)

$$\Phi^* J^{*T} = SN_s A^{-1} J_{t|s}^T, \quad (2.79)$$

$$\left(J^* \Phi^* J^{*T} \right)^{-1} = \Lambda_{t|s}, \quad (2.80)$$

where $\Lambda_{t|s}$ is the support consistent task inertia shown in (2.58). Using the above expressions in the expression of \bar{J}^* shown in (2.81) we obtain the desired equality. \square

Corollary 2.2.4 (Dynamically consistent generalized inverse of J^*). *The following expression is a dynamically consistent generalized inverse of J^**

$$\bar{J}^* \triangleq \Phi^* J^{*T} \left(J^* \Phi^* J^{*T} \right)^{-1}, \quad (2.81)$$

where Φ^* is the constrained projection of A^{-1} as shown in (2.42).

Proof. The task space equation of motion given in (2.59) gives us the relationship between task space forces and joint torques, i.e.

$$\bar{J}_{t|s}^T (SN_s)^T \Gamma = F. \quad (2.82)$$

Using the expression of $\bar{J}_{t|s}$ given in (2.57), the correspondence between $J_{t|s}$ and J^* shown in (2.50), and the equality $\Lambda_{t|s} = \left(J^* \Phi^* J^{*T} \right)^{-1}$ shown in (2.80) the above expression can be written as

$$\bar{J}^{*T} \Gamma = F. \quad (2.83)$$

Considering the whole-body torque vector given in (2.76) and the null expression of Equation (2.78) we can rewrite Γ as

$$\Gamma = J^{*T} F + \left(I - J^{*T} \bar{J}^{*T} \right) \Gamma_0. \quad (2.84)$$

Because \bar{J}^* is a generalized inverse of J^* that cancels the effect of null space torques into task forces and that provides dynamic mapping between joint torques and task forces as shown in (2.83), we will refer to it as the dynamically consistent generalized inverse of J^* . \square

2.3 Control of Internal Forces

When the robot is in double support stance (see Figure 2.6) or when multiple limbs are used for support (e.g. limb contacts during a crawling motion), the close loops formed by the limbs in contact are susceptible to internal forces. Adequate control of the robot's COG or ZMP are necessary but not sufficient conditions to ensure stability of the system over the supporting surfaces. To ensure full stability, the internal forces between supporting limbs need to be controlled as well. For instance, during double support stance there are six internal forces that need to be dealt with. These forces correspond to differential forces and moments on both feet caused by the robot actuation torques. On the other hand, during single support stance, no closed loops are formed and therefore no internal forces appear.

During double support stance, the matrix $SN_s \in \mathcal{R}^{n \times (n+6)}$ is rank deficient, meaning that there are internal displacements —infinitesimally small— that do not contribute to net movement, i.e. internal forces. In fact, the rank of SN_s is $n - 6$ where n is the number of actuated joints, corresponding to the six internal forces that can take place between the supporting feet.

Given the robot's equation of motion shown in (2.24), the torque components associated with internal forces are those that do not contribute to net movement, i.e. components that are filtered with the null space of $(SN_s)^T$, i.e.

$$\left(I - \overline{(SN_s)^T} (SN_s)^T \right) \Gamma_i, \quad (2.85)$$

where Γ_i is a vector of control torques affecting only internal forces. It is straightforward to check that the above term vanishes away when plugged into (2.24).

When combined with the operational space controller presented in (2.76), the complete torque actuation vector including the above torque term takes the form

$$\Gamma = J^{*T} F + N^{*T} \Gamma_0 + \left(I - \overline{(SN_s)^T} (SN_s)^T \right) \Gamma_i. \quad (2.86)$$

Notice that the above term involving internal force control will not produce motion or force coupling at the task point. This can be verified by plugging the term (2.85) into the task's equation of motion (2.59) and verifying that it vanishes away.

Although still under research, using Γ_i we will be able to control the internal forces and moments between supporting limbs. In fact, these forces and moments correspond to

a subset of the reaction forces shown in (2.21), i.e.

$$\{F_i \in \mathcal{R}^6\} \subset \{F_r \in \mathcal{R}^{12}\}, \quad (2.87)$$

where F_i is the vector of internal forces and moments.

In our current implementation we do not yet control internal forces in this fashion. Instead we eliminate torque components that produce internal forces after computing the torque control vector. We are currently in the process of implementing the proposed internal force controller.

Throughout this dissertation we will assume that internal forces are properly controlled and we will focus on the control of operational and postural tasks. Therefore, in general we will use the control structure given in (2.76).

2.4 Examples

Let us study a few examples involving a simulated humanoid model called Stanbot (see Figure 2.9). Stanbot is a robot model with an approximate height of $1.70m$ and a weight of $66kg$. Its body is similar in proportions to the body of an average human. Its articulated body consists of 29 joints, with 6 joints for each leg, 7 for each arm (3 shoulder, 1 elbow, 3 wrist), 1 for the chest yaw movement, and 2 for the head pitch and yaw movements. The masses and inertias of each link have been designed to approximate human equivalent parts.

The purpose of these experiments is to demonstrate how to control the robot's COG. In particular we will demonstrate how to control static balance using the horizontal components of the COG and how to control the posture stance using the vertical component of the COG.

2.4.1 Balance Stability

In Figure 2.9 an experiment involving the control of body balance is shown. Balance is achieved by direct control of the horizontal projection of the COG, characterized by the following coordinate vector and Jacobian matrix,

$$x = \frac{1}{M} \sum_i m_i x_{\text{cogh}_i} \in \mathcal{R}^2, \quad J = \frac{1}{M} \sum_i m_i J_{\text{cogh}_i} \in \mathcal{R}^{2 \times (6+n)}, \quad (2.88)$$

where x_{cogh_i} is the 2×1 COG horizontal vector of the i -th link, and J_{cogh_i} is the associated Jacobian.

The goal here is to track the trajectory shown in the data graphs of Figure 2.9. We use

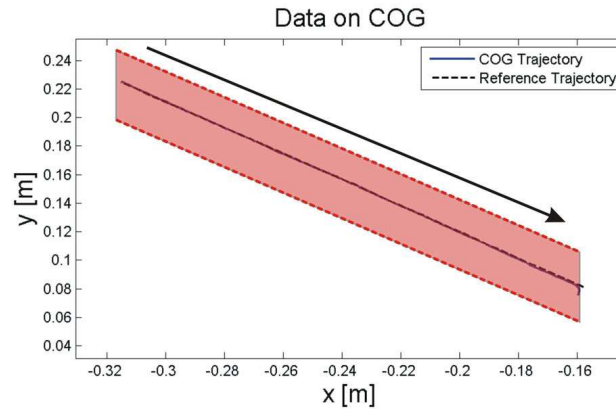
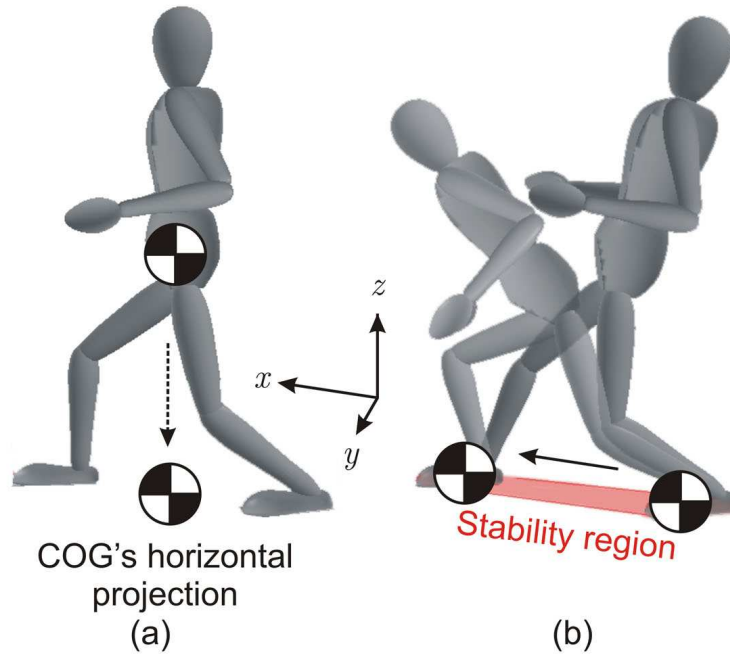


Figure 2.9: **Control of static balance:** In this experiment COG movement was achieved by controlling the horizontal movement of the COG as shown in (a). The COG horizontal position is commanded to follow the trajectory shown in the data graph. Two snapshots of the robot's movement are shown on (b). Using operational space control we can directly control the COG's movement. Because of dynamic compensation, the resulting trajectory follows very closely the desired reference trajectory. The balance stability polygon (i.e. the area defined by supporting feet) is shown in (b) and projected in the data graph.

the following simple PD tracking law

$$a^{ref} = -k_p(x - x_{des}(t)) - k_v\dot{x}, \quad (2.89)$$

where $x_{\text{des}}(t)$ is a desired trajectory within the stability polygon that was designed to not violate ZMP conditions. When applying this control law to the control structure proposed in Equations (2.63) and (2.66), and using the associated linear relationship

$$\ddot{x} = a^{ref}, \tag{2.90}$$

we obtain the following linear behavior of the COG horizontal projection

$$\ddot{x} + k_v \dot{x} + k_p (x - x_{\text{des}}(t)) = 0. \tag{2.91}$$

Here we have not addressed the control of the residual redundancy. In fact, for this experiment residual DOFs are used to maintain an upright posture using the methods we will describe in Chapter 4.

2.4.2 COG Vertical Control

A second experiment is shown in Figure 2.10, where both horizontal and vertical components of the COG are controlled. While the horizontal components of the COG are used to keep static balance as shown in the previous example, the vertical component is used to control posture stance.

This time the control coordinates are

$$x = \begin{pmatrix} x_{\text{cogh}} \\ x_{\text{cogv}} \end{pmatrix}, \tag{2.92}$$

where x_{cogh} corresponds to the 2×1 horizontal component of the COG and x_{cogv} corresponds to the 1×1 vertical component. The acceleration level control vector can be broken into vertical and horizontal components, i.e.

$$a^{ref} = \begin{pmatrix} a_{\text{cogh}}^{ref} \\ a_{\text{cogv}}^{ref} \end{pmatrix}. \tag{2.93}$$

The control law for the horizontal component is the same as in (2.89), whereas the control law for the vertical component is

$$a_{\text{cogv}}^{ref} = -k_p (x_{\text{cogv}} - x_{\text{des}}(t)) - k_v \dot{x}_{\text{cogv}}. \tag{2.94}$$

However, this time the reference trajectory for balancing is fixed at a point within

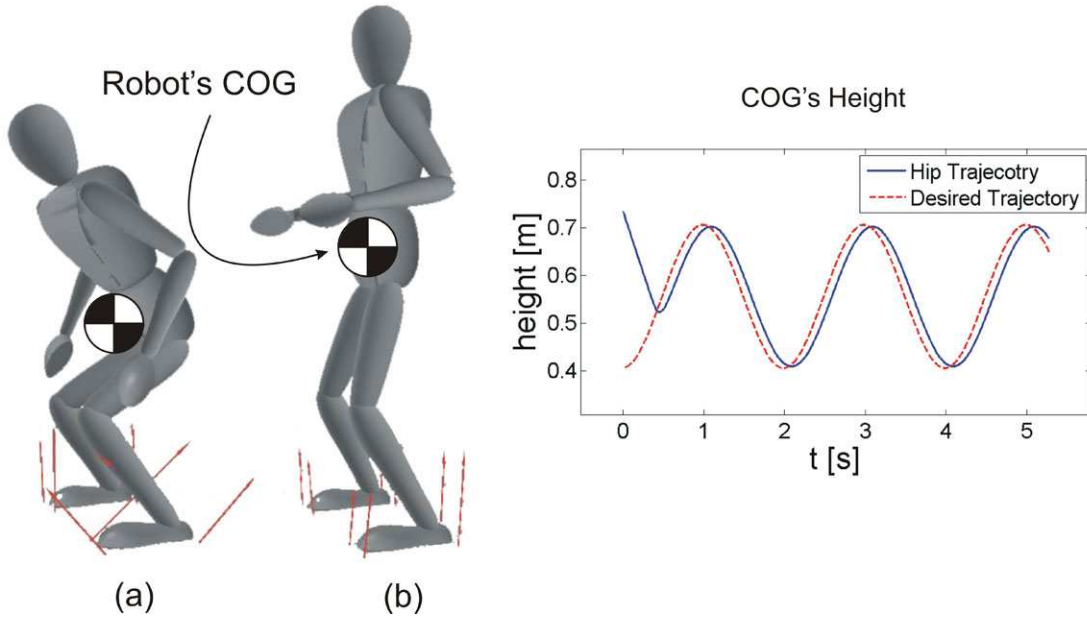


Figure 2.10: **Control of the COG's height:** This experiment demonstrates the control of the COG's height while maintaining balance. The COG's vertical position is controlled to track a sinusoidal trajectory spanning positions (a) and (b).

the feet stability polygon while the reference trajectory for the vertical component is the sinusoidal trajectory shown in Figure 2.10.

The proportional gain k_p used for the experiment is equal to $400N/m$ and the velocity gain is chosen to provide critical damping, i.e. $k_v = 2\sqrt{k_p}$. The results shown in the data graph of Figure 2.10 proof a good tracking response.

Chapter 3

Prioritized Multi-Task Control

In this chapter we will develop control structures for the simultaneous execution of multiple operational tasks using prioritized controls strategies. These structures will allow us to create complex whole-body behaviors while ensuring that critical tasks are accomplished first.

Current approaches for the control of humanoid systems address the control of manipulation and locomotion behaviors as separate entities, disregarding their combined interactions. While these approaches are practical they do not exploit the redundancy of the system to simultaneously accomplish all required objectives. A few projects have addressed whole-body control, most notably the work by the National Institute of Advanced Industrial Science and Technology in Japan (see Neo et al. 2005). The main objective of this chapter is to design methods to control collections of multiple low-level tasks towards individual goals using whole-body movements and while ensuring that critical tasks are accomplished first. For instance, for locomotion behaviors separate goals are chosen for feet and COG or ZMP placement while a whole-body controller automatically assigns torque resources to accomplish all goals. These goals can be fed at runtime by a sensory layer without pre-computation of feet or COG trajectories. To ensure stability, critical tasks need to be first controlled. For instance in the previous example, COG control and feet placement control operate as priority tasks, while other less critical tasks operate in the residual movement redundancy.

Task prioritization was first addressed by (Hanafusa, Yoshikawa, and Nakamura 1981) and will be extensively addressed here. Multi-task control in the context of manipulation was first addressed by (Siciliano and Slotine 1991). Similar control structures were later developed for graphical models of humanoid systems by (Boulic and Mas 1996) and later by (Baerlocher and Boulic 1998). All of these preliminary research was developed in the

context of inverse kinematic control.

Instead of relying on inverse kinematic representations, we will develop here operational space control representations. Operational space control will allow us to integrate force, position, and impedance control policies into a unified formulation. Additionally, we will integrate task prioritization to decouple task movement and to solve conflicting scenarios as a central part of our operational space control framework. Two indexes will be proposed to measure task feasibility, the condition number of constrained task Jacobians and the condition number of constrained task inertias. We will use task feasibility measurements to modify or replan robot behavior at runtime.

A problem associated with most humanoid control platforms is their limited ability to engage in advanced contact interactions. Some progress has been done in this area with a handful of applications available and customized for very specific scenarios (see Yokoyama et al. 2003 and Harada et al. 2004). However, more progress in this area is needed addressing multi-contact and human-like interactions. Past approaches on contact control rely on local inverse kinematic representations which map desired forces into joint trajectories. However, little consideration has been given to the impact of contacts on the overall robot behavior. In the previous chapter we presented methods to integrate whole-body torque control with contact constraints and we studied the control of multiple non-prioritized tasks within the constrained space. In this chapter we will extend this approach to the control of multiple prioritized tasks involving manipulation and locomotion behaviors. Each task will be controlled within the residual redundancy of supporting constraints and higher priority tasks as part of a prioritized control strategy, providing decoupling effects and a platform to resolve conflicting scenarios. At the same time, supporting constraints will be automatically accounted for. Additionally, we will design operational space controllers for each individual task, allowing the user to implement a variety of advanced control strategies for manipulation, locomotion, and postural behaviors. In our approach contact tasks will not be limited to end effector control. Arbitrary parts on the robot's body will be able to act as support or contact points. For instance, for sitting tasks, the points below the hips will be used for support. In essence the approaches we will present here will allow humanoids to simultaneously accomplished multiple prioritized goals without relying on trajectory generation.

To cope with the uncertainty associated with realtime interactions, the whole-body control framework we will develop throughout this chapter will be based on task hierarchies. In these hierarchies, the control of lower priority tasks will be conditional to the accomplishment of higher priority tasks. This organization will provide decoupling effects between

tasks and will allow controllers to determine the feasibility of the overall behavior at runtime. For instance, balance control will determine the feasibility of lower priority tasks such as manipulation and locomotion.

This chapter is organized as follows. In Section 3.1 we will study the composition of whole-body behaviors as aggregations of multiple goals. We will discuss prioritization to ensure that critical tasks are accomplished first and we will develop kinematic representations based on such hierarchies. In Section 3.2 we will develop operational space control representations to simultaneously control multiple task points. We will develop mathematics for identifying the residual movement redundancy for each priority level and we will discuss control policies to implement a variety of low-level behaviors. Finally, in Section 3.3 we will present several examples involving interactive manipulation and locomotion behaviors.

3.1 Representation of Whole-Body Behaviors

Although the goal of humanoid systems is to execute advanced locomotion and manipulation tasks, other low-level tasks need to be simultaneously accomplished to support and enhance the overall behavior. To create complex behaviors we must first understand the contribution of different body parts to the overall movement. In this section we will analyze the composition of complex whole-body behaviors and their representation at the kinematic level.

As discussed in the previous chapter, balance and internal forces between supporting limbs are first controlled to provide overall stability. Manipulation and locomotion tasks need to be integrated without interfering with these and other more critical tasks. Control prioritization will allow us to create this control separation. Once more our approach will be to create methods for the realtime control of goal-based behaviors instead of relying on pre-programmed trajectory-based control methods. For instance, for manipulation tasks we will consider torque controllers to accomplish desired hand position and orientation goals while for locomotion tasks we will consider torque controllers to accomplish desired feet position and orientation as well as desired COG and ZMP positions or trajectories. In turn, to create simultaneous locomotion and manipulation behaviors, only a few control points will need to be actively controlled while the rest of the robot's movement will be automatically generated to comply with contact constraints and postural criteria.

By controlling individual task points we will be able to create complex manipulation and locomotion behaviors in realtime. For each task point, we will associate an operational space controller implementing a specific control policy. Desired position or force goals will

be either fed by the sensory layer at runtime or pre-programmed to fulfill predetermined rules or trajectories. For instance for manipulation tasks, the goal position and orientation of the robot's hands could be fed at runtime by the robot's vision system. At the same time, balance stability and postural behaviors will be simultaneously controlled according to rules or trajectories. The whole-body torque controller we will describe here will be able to simultaneously accomplish all desired goals while ensuring that critical tasks are accomplished first. By exploiting the redundancy within our hierarchical model our controller will automatically derive appropriate torque solutions while providing movement stability at all times.

3.1.1 Multi-Task Decomposition

Multi-task control is our approach to simplify the synthesis of whole-body behaviors. When a behavior is sought, the individual actions that need to take place are first determined. For example for a walking behavior, each phase involving a different supporting leg is defined as a separate movement. The desired behavior emerges by sequencing the movements. In this context individual movements are units of action where each action corresponds to a fixed set of tasks simultaneously operated towards individual goals. For instance for a movement designed to step forward, the operational points that need to be controlled are the position and orientation of the swinging leg, the global COG, and the head orientation. When the swinging leg makes contact with the ground, the tasks involved in the previous stepping motion are discarded and a new set of tasks is used to implement the next phase. Our focus here is on the individual movements, or equivalently on the set of tasks needed to achieve the desired behaviors.

To study the synthesis of complex behaviors through multi-task decomposition let us consider the interactive behavior shown in Figure 3.1. Here the objective is to place screws at desired locations in the wooden beams, simulating insertion with a screwgun. This behavior can be synthesized in realtime by controlling four separate operational tasks and one postural task, each controlling a different aspect of the robot's movement as shown in Table 3.1. Moreover the two feet in contact with the ground provide the support for balance stability. To guarantee feet stability, internal forces between legs are controlled to vanish or to maintain the feet flat against the ground. For balance stability, the robot's COG horizontal position is controlled to stay above the feet stability polygon. Hand teleoperation is achieved by controlling the 6-D spatial position of the hand. The teleoperated reference point is shown as a small red sphere located at the center of the right hand in Figure 3.1. Head orientation is achieved by controlling the two orientation coordinates associated with

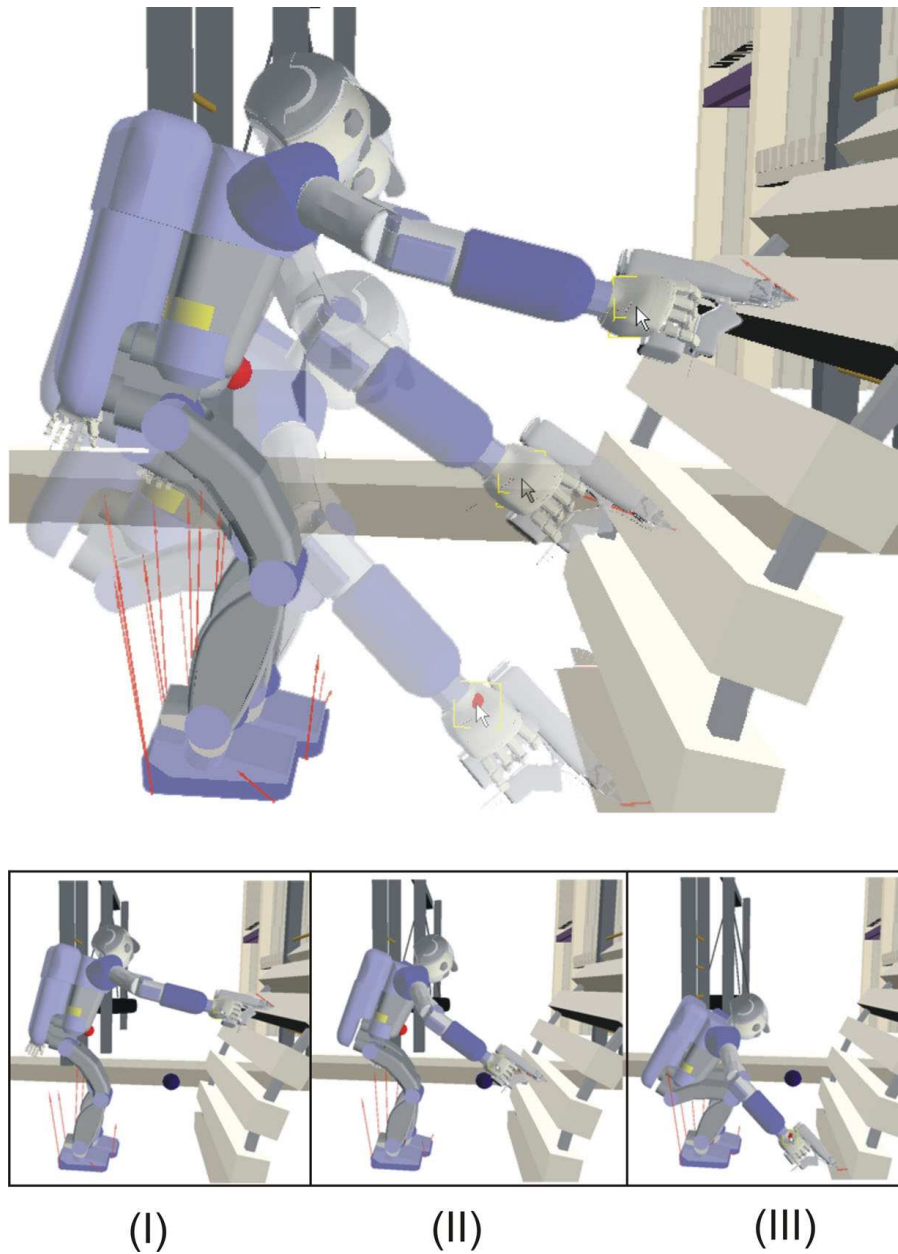


Figure 3.1: **Manipulation behavior with screwgun:** This sequence of images correspond to an interactive manipulation behavior. A screwgun is teleoperated to insert screws into the wooden beams. All other aspects of the motion are automatically handled. In particular, the robot's posture is based on two distinct snapshots of human poses that will be later discussed and a switching policy between postures is implemented.

the robot’s gaze (i.e. the ray emerging forward from the head on the direction of the eyes). In our example the desired orientation corresponds to aligning the robot’s gaze with the teleoperated point. Notice that both the robot’s right hand and its head are commanded to track in realtime the teleoperated point. Although the function of the tasks discussed for the above example is straightforward, the posture behavior involves perhaps the most complex control. We control it to mimic human poses. Posture control will be discussed in great detail in Chapter 4.

Task Decomposition (Screwgun Manipulation)

<i>Task Primitive</i>	<i>Coordinates</i>	<i>DOFs</i>	<i>Control Policy</i>
Balance	COG horizontal position	2 ($x - y$ plane)	position
Manipulation	right hand pos/ori	6	hybrid
Gaze	head orientation	2 (\perp plane)	position
Posture	whole-body joints	$n = \text{NumJoints}$	optimal criterion

Table 3.1: **Task decomposition for the manipulation behavior of Figure 3.1.**

In this chapter we will develop operational space control methods for controlling multiple task points based on a prioritized control strategy between tasks. Operational space control provides direct control of task forces and accelerations, facilitating the implementation of a variety of control strategies including force, position, acceleration, and impedance control. In the last column of Table 3.1 we indicate the control policy that we have implemented for each task primitive of the previous example. For instance, balance control is achieved through PD control of the COG’s position, right hand control is achieved through hybrid position and force control of the tool tip, head orientation control is based on PD control of the head’s orientation, and posture control is achieved through gradient projection of optimal criteria. These control strategies will be discussed in more detail at the end of this chapter.

Our approach for task control is based on the realtime command of desired goals without relying on trajectory computation. Goals are directly mapped into task forces or accelerations and further converted into actuation torques using operational space controllers. These goals are issued from a variety of sources, such as from predefined rules, teleoperation devices, or sensors. For instance, in the above example balance is achieved by controlling the COG’s position towards the center of the robot’s feet, hand and head movements are

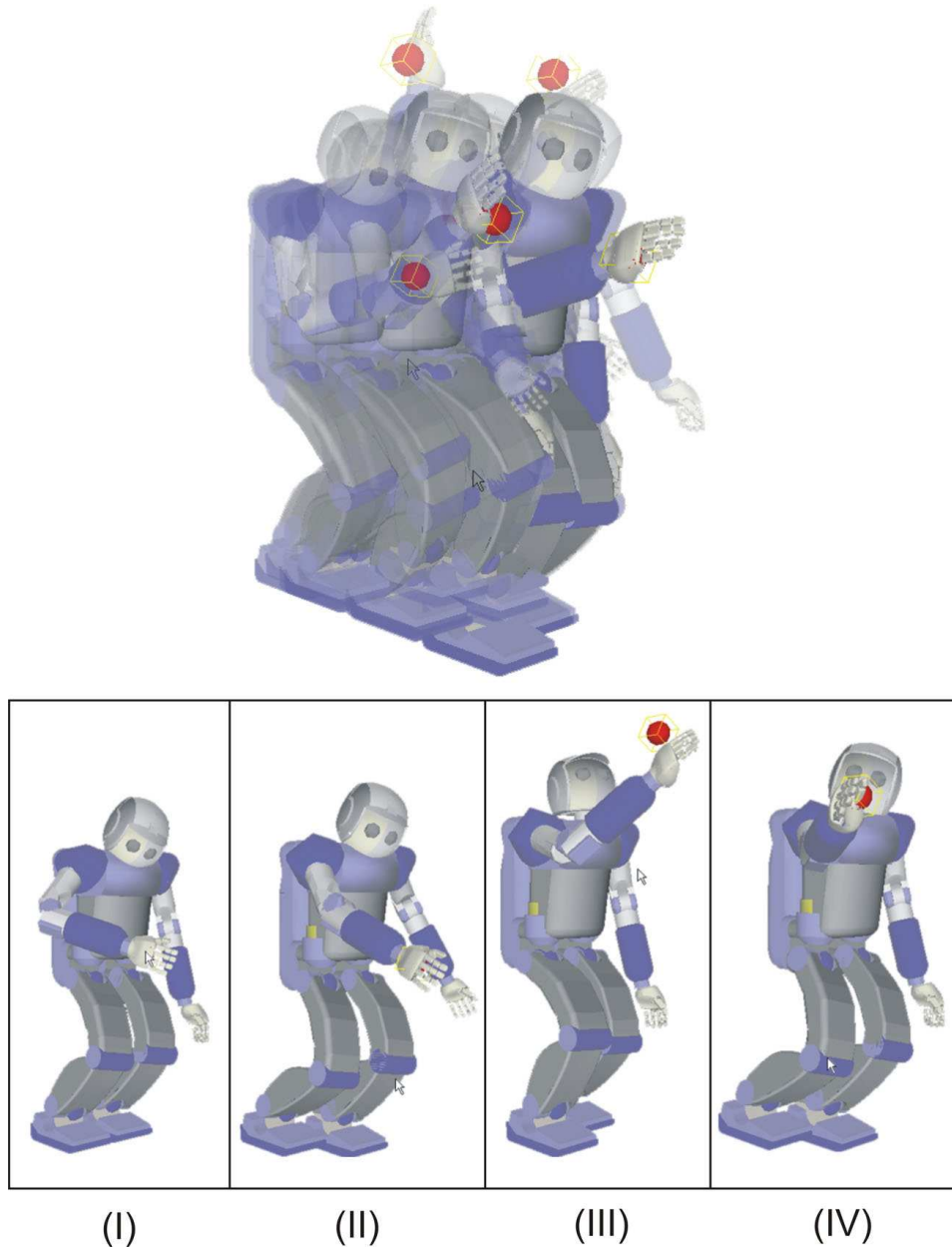


Figure 3.2: **Simultaneous locomotion and manipulation behavior:** In this example, the goal is to walk forward while tracking a teleoperated point. The sequence of images above correspond to an actual simulation where the walking pattern is pre-programmed and the hand is teleoperated. The red sphere corresponds to the desired teleoperated positions. (I), (II), and (III) are snapshots taken during movement execution.

Task Decomposition (Stepping Forward with Hand Teleoperation)

<i>Task Primitive</i>	<i>Coordinates</i>	<i>DOFs</i>	<i>Control Policy</i>
Balance	COG horizontal position	2 ($x - y$ plane)	position
R Foot placement	foot pos/ori	6	position
Gaze orientation	head orientation	2 (\perp plane)	position
Hip height	hip vertical position	1 (z-axis)	position
Hip posture	hip orientation	3	position
Chest posture	chest yaw orientation	1 (z-axis)	position
R hand Manipulation	right hand pos/ori	6	position
R shoulder roll posture	right shoulder roll	1	position
L arm posture	left arm joints	7	position

Table 3.2: **Task decomposition for the walking example shown in Figure 3.2.**

commanded to track the teleoperated point, and posture motion is commanded to minimize the joint space error with respect to captured human poses. While some aspects of the motion, such as balance, internal forces, and posture stance are automatically handled by pre-defined rules, others such as manipulation and locomotion are interactively handled. Minimizing the amount of points interactively controlled simplifies the creation of complex behaviors. For instance, in the previous example only one external command is needed to create the desired tool movement, corresponding to the position and orientation of the teleoperated point, while all other aspects of the motion are automatically handled.

We consider a second more complex behavior involving simultaneous locomotion and manipulation behaviors as shown in Figure 3.2. This example corresponds to an actual simulation of walking while the robot’s right hand is teleoperated by a user. Walking is created by sequencing four movements: (1) shifting the robot’s weight to the right foot, (2) swinging the right foot forward, (3) shifting the weight back to the left foot, and (4) swinging the left foot forward. Additionally, the head orientation and the right hand position are controlled to track a teleoperated point (shown as a red sphere). For a more in-depth study on sequencing whole-body movements refer to Chapter 7. Table 3.2 depicts the task decomposition we used for the walking phase involving swinging the right foot forward.

3.1.2 Constrained Kinematics

Task kinematics under supporting constraints were analyzed in the previous chapter in the context of non-prioritized control. The associated constrained Jacobian allowed us to characterize task feasibility under multi-leg support and to derive operational space controllers in a compact form.

When multiple operational points are simultaneously controlled as part of a whole-body behavior, each point can be kinematically characterized through unique constrained Jacobians. For a given behavior (e.g. the behaviors of Figures 3.1 or 3.2), the full kinematic representation of an arbitrary task point k (see Tables 3.1 and 3.2 for a list of tasks) is

$$x_k = \begin{pmatrix} x_{k,p} \\ x_{k,r} \end{pmatrix}, \quad (3.1)$$

where $x_{k,p}$ is a position representation of the task point and $x_{k,r}$ is an orientation representation. Position and orientation representations can vary depending on the type of task being implemented as discussed in (Khatib 2004). Moreover, a subset of these coordinates can be considered for tasks involving fewer than 6 DOFs.

Task velocities can be expressed using arbitrary representations by considering linear and angular velocities of the task point and transforming them to the desired representation, i.e.

$$\dot{x}_k = E(x_k) \begin{pmatrix} v_k \\ \omega_k \end{pmatrix}, \quad (3.2)$$

where $E(x_k)$ is a representation transformation matrix and is described in (Khatib 2004) and v_k and ω_k are linear and angular velocities of the k -th task point respectively. The instantaneous kinematics of arbitrary task points is expressed in terms of base and joint velocities as

$$\dot{x}_k = J_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad J_k = E(x_k) J_{k,0}, \quad (3.3)$$

where J_k is the task Jacobian in task coordinates and can be derived from the basic task Jacobian $J_{k,0}$ as explained in (Khatib 2004) and ϑ_b and \dot{q} are base and joint velocities respectively.

Definition 3.1.1 (Support consistent Jacobian of arbitrary points). *The following kinematic representation is a generalization of the constrained terms (2.46) and (2.47)*

$$\dot{x}_k = J_k^* \dot{q}^*, \quad (3.4)$$

where

$$J_k^* \triangleq J_k \overline{SN}_s \quad (3.5)$$

is the constrained Jacobian matrix of an arbitrary task point k and \dot{q}^* is the vector of constrained joint velocities shown in (2.37).

As discussed in the previous chapter, the constrained Jacobian J_k^* characterizes the kinematic behavior of task points under structural constraints and supporting contacts. For instance, if legs, upper body, and arms are fully stretched, the constrained Jacobian associated with hand operational points will become singular due to supporting constraints acting on the underactuated system.

More complete kinematic representations should also integrate constraints imposed by task hierarchies. In the next section we will introduce the concept of prioritization to integrate additional constraints at the kinematic and dynamic levels, allowing us to determine if lower priority tasks can be accomplished in the null space of higher priority tasks.

3.2 Control Structures

When controlling multiple task points towards arbitrary goals, coupling effects between tasks can arise due to modeling errors and conflicts between task goals. Though these problems could be solved through motion planning techniques, we consider here alternative reactive methods. Our approach consists on establishing a hierarchy between tasks where lower priority tasks are projected in the null space of higher priority tasks. These projections not only provide the desired decoupling effects but also ensure that higher priority tasks are first accomplished.

3.2.1 Prioritization Versus Aggregation

Let us consider an arbitrary behavior (e.g. the screwgun behavior of Table 3.1). Each k -th operational point can be represented by the coordinate vector

$$x_k \in \mathcal{R}^{\dim(k)}, \quad (3.6)$$

where x_k is normally a subset of the 6D cartesian position and orientation of the k -th task point as shown in (3.1) — although for some tasks x_k could be a subset of the joint coordinates of the robot (e.g. for tasks involving stretching the knees) — and $\dim(k)$ is the dimension of the task.

In the previous chapter, we suggested that we could simultaneously control all tasks by considering a non-prioritized macro task characterized by the following multi-point coordinate vector and Jacobian matrix

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}, \quad J = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_N \end{pmatrix}, \quad (3.7)$$

where x_i and J_i correspond to the coordinates and full Jacobian of the i -th task point respectively. The associated control vector had the form shown in (2.64), i.e.

$$\Gamma = J^{*T} F, \quad (3.8)$$

where $J^* = \overline{J\mathcal{S}\mathcal{N}_s}$ is the constrained Jacobian matrix shown in (2.47) and F is a force-level control vector determining control policies for all task points.

Although aggregating non-prioritized tasks into a single macro structure is an attractive solution it entails coupling problems between tasks as mentioned earlier. For instance, in the behavior shown in Figure 3.2, motion of the right hand will normally result in coupling effects on locomotion tasks due to modeling errors. In particular, these coupling effects will affect the trajectories established for the robot's COG and the swinging foot. However, COG stability and feet motion are critical to the overall stability of the robot. Therefore coupling effects on these tasks should be removed.

Our solution to the above coupling problem is to establish a task hierarchy by exploiting the prioritized structure presented in Equation (2.76) of the previous chapter, which can filter out coupling components introduced by lower priority tasks. For instance, for a two-level prioritized structure — e.g. the simultaneous control of COG and hand positions where the COG is considered as a high priority task — we associate the following control structure

$$\Gamma = J_1^{*T} F_1 + N_1^{*T} \Gamma_2, \quad (3.9)$$

where $J_1^* = \overline{J_1\mathcal{S}\mathcal{N}_s}$ is the constrained Jacobian of the priority task (e.g. the COG's position)

as shown in (2.47), F_1 is the associated vector of control forces,

$$N_1^* \triangleq I - \bar{J}_1^* J_1^* \quad (3.10)$$

is the associated constrained null space matrix and is equivalent to (2.78), and Γ_2 is a vector of torques associated with the control of the secondary task (e.g. the control of the hand) within the residual redundancy.

For more complex behaviors, multiple priority levels can be established. For instance, the behavior shown in Figure 3.4 could be created using the prioritized hierarchy shown in Table 3.3. This hierarchy is established according to the relative importance of each task.

Hierarchy (Locomotion and Hand Control)

<i>Task Primitive</i>	<i>Priority Level</i>
Balance	1
R foot placement	2
Gaze orientation	3
Hip height	4
Hip posture	5
Chest posture	6
R hand Manipulation	7
R shoulder roll posture	8
L arm posture	9

Table 3.3: **Prioritized hierarchy for the behavior shown in Figure 3.4**

Notice that right hand control is considered a lower priority task than postural tasks. This ordering reflects the importance of postural motion on the overall stability of the walk. Also notice that feet placement control has been assigned the second highest priority level after COG control. Once more, this assignment reflects the relative importance of feet placement towards the overall stability of the walk. In the previous example, the proposed hierarchy has been empirically established.

In general for N arbitrary prioritized tasks we propose the following recursive whole-body torque control structure

$$\Gamma = \Gamma_1 + \Gamma_{2|\text{prec}(2)} + \cdots + \Gamma_{N|\text{prec}(N)} = \sum_{k=1}^N \Gamma_{k|\text{prec}(k)}, \quad (3.11)$$

where the subscript $k|\text{prec}(k)$ is used to indicate that the k -th task operates in the null space of all higher priority tasks.

Definition 3.2.1 (Prioritized torques). *The following expression determines the projection of lower priority tasks into the null space of higher priority tasks*

$$\Gamma_{k|\text{prec}(k)} \triangleq N_{\text{prec}(k)}^T \Gamma_k, \quad (3.12)$$

where $N_{\text{prec}(k)}$ is the combined null space of all higher priority tasks (i.e. all preceding tasks) to the k -th level and will be characterized in a few lines.

Based on the previous hierarchical torque control structure (3.11), we will be able to formulate a general operational space control structure that will take the form

$$\Gamma = \left(J_1^{*T} F_1 \right) + \left(J_{2|1}^{*T} F_{2|1} \right) + \cdots + \left(J_{N|\text{prec}(N)}^{*T} F_{N|\text{prec}(N)} \right), \quad (3.13)$$

where the matrices $J_{k|\text{prec}(k)}^*$ correspond to prioritized task Jacobians as defined in (3.14), and the vectors $F_{k|\text{prec}(k)}$ correspond to control forces to control the k -th priority task. These recursive structures will be demonstrated in the next section.

Definition 3.2.2 (Prioritized Jacobian). *The following prioritized Jacobian is associated with the k -th priority task*

$$J_{k|\text{prec}(k)}^* \triangleq J_k^* N_{\text{prec}(k)}^*, \quad (3.14)$$

where J_k^* is the constrained Jacobian associated with the k -th task as shown in (3.5) and $N_{\text{prec}(k)}^*$ is the prioritizing null space of all preceding tasks and will be characterized in (3.40).

The second objective of projecting lower priority tasks in the null space of higher priority tasks is to solve conflicting scenarios when multiple goals cannot be simultaneously accomplished. For instance, let us consider the interactive behavior shown in Figure 3.3.

Here, the robot's hand is commanded to move towards unreachable goals. When the

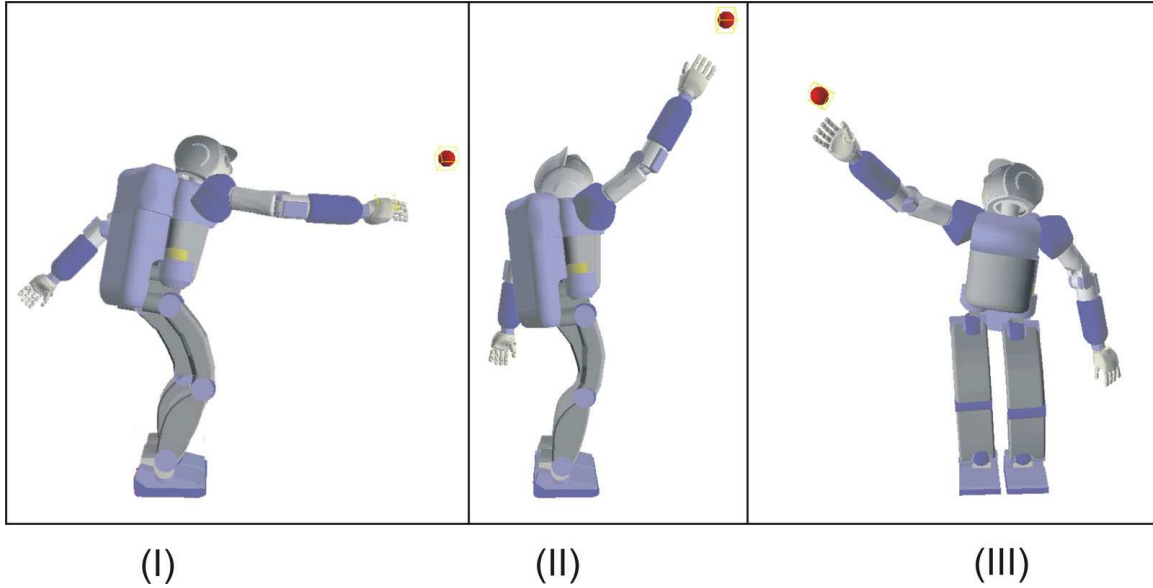


Figure 3.3: **Conflict between task goals:** These snapshots corresponding to a teleoperated reaching behavior illustrates potential conflicts between task goals. The right hand is commanded to reach the red sphere which is intentionally placed beyond the reachable workspace at different locations. In (I) and (III), the red sphere cannot be reached without compromising balance stability. In (II) the sphere cannot be reached without jumping.

hand approaches these goals, balance stability is compromised. If COG control is given higher priority than hand control, we will not only prevent coupling effects between COG and hand control, but we will ensure that balance goals are first accomplished while hand goals are accomplished only if there is enough available residual redundancy. In general, the hierarchy established in Table 3.3 will determine whether lower priority tasks can be executed without compromising higher priority tasks.

In fact, the prioritized Jacobian of Equation (3.14) will reveal not only singularities due to supporting constraints but also due to constraints imposed by the task hierarchy. If no movement redundancy is available to execute the k -th priority task within the residual redundancy, the associated prioritized Jacobian will become singular. In contrast, the non-prioritized Jacobian J_k^* will not reflect prioritization singularities. We will exploit this behavior to measure task feasibility at runtime and to modify robot behavior if needed.

In general a combination of prioritized (2.34) and non-prioritized (3.13) control strategies will be simultaneously implemented. For instance, for a dual hand manipulation task, the right and left hand tasks could be combined into a single non-prioritized macro task and then prioritized with respect to a balance task.

Although prioritization has many advantages, the implementation of force and position control policies in the null space of priority tasks is a difficult problem that will be solved in this chapter.

3.2.2 Prioritized Dynamics and Control

Given a multi-task control scenario, we will derive here control structures to simultaneously accomplish all goals. For instance, for the manipulation behavior shown in Figure 3.1 we choose the following task hierarchy,

Task Decomposition (Screwgun Teleoperation)

<i>Task Primitive</i>	Priority level
Balance	1
Gaze orientation	2
Right hand control	3
Whole-body posture	4

The main goal of this hierarchy is to decouple balance control from hand control and to prevent conflicts between these two tasks.

In general, there will be N arbitrary task points to be controlled (for the previous behavior $N = 3$) as well as additional postural tasks. The objective of our controller is to provide linear control of task forces and accelerations while operating in the null space of higher priority tasks.

We consider implementing the prioritized control strategy shown in (3.11) and (3.12) to handle an arbitrary number of tasks. We must first understand the dynamic behavior of each task under the proposed prioritized control structure. In (2.24) we had derived the following equation of motion under supporting constraints,

$$A \begin{pmatrix} \dot{v}_b \\ \ddot{q} \end{pmatrix} + N_s^T(b + g) + J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \dot{b} \\ \dot{q} \end{pmatrix} = (SN_s)^T \sum_{i=1}^N \Gamma_{i|\text{prec}(i)}, \quad (3.15)$$

where N_s is the dynamically consistent null-space matrix of the Jacobian at the support points (2.25), and $S = \begin{pmatrix} 0_{n \times 6} & I_{n \times n} \end{pmatrix}$ is the actuation matrix discussed in (2.19). Notice that in the above equation we have directly used the prioritized torque control structure proposed in (3.11).

We can derived task dynamics by left-multiplying (3.15) by the term $J_k A^{-1}$, where J_k

is the full Jacobian of the k -th operational task yielding the following equation of motion

$$\ddot{x}_k - \dot{J}_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + J_k A^{-1} N_s^T (b + g) + J_k A^{-1} J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = J_k A^{-1} (S N_s)^T \left(\Gamma_{k|\text{prec}(k)} + \sum_{(i>0) \wedge (i \neq k)}^N \Gamma_{i|\text{prec}(i)} \right). \quad (3.16)$$

Here we have used the equality

$$\ddot{x}_k = J_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + \dot{J}_k \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} \quad (3.17)$$

and we have decomposed the actuation torques into torques allocated to control the k -th operational task and torques allocated to control all other operational tasks, i.e.

$$\sum_{i=1}^N \Gamma_{i|\text{prec}(i)} = \left(\Gamma_{k|\text{prec}(k)} + \sum_{(i>0) \wedge (i \neq k)}^N \Gamma_{i|\text{prec}(i)} \right). \quad (3.18)$$

Definition 3.2.3 (Prioritized inertia). *The following term is referred to as the prioritized inertia of the k -th priority task*

$$\Lambda_{k|\text{prec}(k)}^* \triangleq \left(J_{k|\text{prec}(k)}^* \Phi^* J_{k|\text{prec}(k)}^{*T} \right)^{-1}. \quad (3.19)$$

Theorem 3.2.1 (Prioritized operational space control). *The following control vector yields linear control of forces and accelerations for the k -th prioritized task*

$$\Gamma_{k|\text{prec}(k)} = J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)}, \quad (3.20)$$

where $\Gamma_{k|\text{prec}(k)}$ is the k -th component of the prioritized torque control structure shown in (3.11), $J_{k|\text{prec}(k)}^*$ is the prioritized Jacobian for the k -th task point discussed in (3.14), and $F_{k|\text{prec}(k)}$ is a vector of control forces that will be discussed in a few lines.

Proof. Based on the above control term, (3.16) becomes

$$\ddot{x}_k - \dot{J}_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + J_k A^{-1} N_s^T (b + g) + J_k A^{-1} J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = \left(\Lambda_{k|\text{prec}(k)}^* \right)^{-1} F_{k|\text{prec}(k)} + J_k A^{-1} (S N_s)^T \sum_{(i>0) \wedge (i \neq k)}^N \Gamma_{i|\text{prec}(i)}, \quad (3.21)$$

where the term $\Lambda_{k|\text{prec}(k)}^*$ is the inertial term defined in (3.19) whose inverse maps prioritized forces into task accelerations and fulfill the following equality as demonstrated in Property 3.2.1

$$\left(\Lambda_{k|\text{prec}(k)}^* \right)^{-1} = J_k A^{-1} (S N_s)^T J_{k|\text{prec}(k)}^{*T}. \quad (3.22)$$

Under normal conditions $\Lambda_{k|\text{prec}(k)}^*$ is full rank, and therefore the vector $F_{k|\text{prec}(k)}$ yields linear control of task accelerations and forces. \square

Property 3.2.1 (Alternative expression of $\Lambda_{k|\text{prec}(k)}^*$). *The prioritized inertia defined in (3.19) has the alternative expression given in (3.22).*

Proof. Using the expression of $J_{k|\text{prec}(k)}^*$ given in (3.14), the property $N_{\text{prec}(k)}^* \Phi^* = \Phi^* N_{\text{prec}(k)}^{*T}$ shown in (A.15), and the property $(N_{\text{prec}(k)}^*)^2 = N_{\text{prec}(k)}^*$ shown in (A.10) the following equality holds

$$J_{k|\text{prec}(k)}^* \Phi^* J_{k|\text{prec}(k)}^{*T} = J_k^* \Phi^* J_{k|\text{prec}(k)}^{*T}. \quad (3.23)$$

Using the expression of J_k^* given in (3.5) and the expression of Φ^* given in (2.42) the above expression becomes

$$J_k \overline{S N_s} S N_s A^{-1} (S N_s)^T J_{k|\text{prec}(k)}^{*T}. \quad (3.24)$$

Using the equality $\overline{S N_s} S N_s = N_s$ given in (2.44), the equality $N_s A^{-1} = A^{-1} N_s^T$ given in (2.27), and the property $(N_s)^2 = N_s$ given in (2.26) the above expression becomes

$$J_k A^{-1} (S N_s)^T J_{k|\text{prec}(k)}^{*T} \quad (3.25)$$

which is equal to (3.19). \square

Using the control expression (3.20) we can easily implement linear control strategies. For instance, to control task accelerations we consider the following force control vector:

Corollary 3.2.1 (Prioritized motion control). *In the absence of contact interactions, the following control vector yields linear control of task accelerations*

$$F_{k|\text{prec}(k)} = \Lambda_{k|\text{prec}(k)}^* a_k^{\text{ref}} + \mu_{k|\text{prec}(k)}^* + p_{k|\text{prec}(k)}^* - \Lambda_{k|\text{prec}(k)}^* J_k A^{-1} (S N_s)^T \sum_{i=1}^{k-1} \Gamma_{i|\text{prec}(i)}. \quad (3.26)$$

Here $F_{k|\text{prec}(k)}$ is the control force shown in (3.20), a_k^{ref} is an acceleration-level control policy for the k -th priority task, and the remaining dynamic quantities for the above equation have the following expressions,

$$\mu_{k|\text{prec}(k)}^* \triangleq \Lambda_{k|\text{prec}(k)}^* J_k A^{-1} N_s^T b - \Lambda_{k|\text{prec}(k)}^* \dot{J}_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + \quad (3.27)$$

$$\Lambda_{k|\text{prec}(k)}^* J_k A^{-1} J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix},$$

$$p_{k|\text{prec}(k)}^* \triangleq \Lambda_{k|\text{prec}(k)}^* J_k A^{-1} N_s^T g. \quad (3.28)$$

Proof. It is straightforward to verify that using the above expressions in (3.21) will yield the linear behavior

$$\ddot{x}_k = a_k^{\text{ref}}. \quad (3.29)$$

Notice that the last term on the RHS of (3.26) disregards lower priority torques. The reason is that prioritized null space matrices are designed to cancel components from lower priority tasks as we will see when demonstrating (3.40). \square

Corollary 3.2.2 (Prioritized multi-task control structure). *The following control structure yields linear control of forces and accelerations of a set of N prioritized tasks*

$$\Gamma = \left(J_1^{*T} F_1 \right) + \left(J_{2|1}^{*T} F_{2|1} \right) + \cdots + \left(J_{N|\text{prec}(N)}^{*T} F_{N|\text{prec}(N)} \right) = \sum_{k=1}^N J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)}. \quad (3.30)$$

Proof. Using (3.20) we can further express (3.11) as the above aggregation of prioritized operational space control structures. \square

3.2.3 Recursive Redundancy

In (3.12), prioritization was established by projecting lower priority tasks within the residual redundancy (i.e. the null-space) of higher priority tasks. In this context, null space

projections impose that lower priority tasks do not introduce acceleration and force components in higher priority tasks. Given Equation (3.16) this is equivalent to the following condition

$$\forall i \in \text{prec}(k) \quad J_i A^{-1} (S N_s)^T N_{\text{prec}(k)}^{*T} = 0. \quad (3.31)$$

Corollary 3.2.3 (Dynamically consistent prioritized null space matrix). *The following matrix fulfills the above set of dynamic constraints*

$$N_{\text{prec}(k)}^{*T} = \prod_{i=1}^{k-1} N_{i|\text{prec}(i)}^{*T}, \quad (3.32)$$

with

$$N_{i|\text{prec}(i)}^* \triangleq I - \overline{J}_{i|\text{prec}(i)}^* J_{i|\text{prec}(i)}^*, \quad (3.33)$$

where $\overline{J}_{i|\text{prec}(i)}^*$ is the dynamically consistent generalized inverse of $J_{i|\text{prec}(i)}^*$ shown in (3.36). Moreover, we use the conventions $N_{\text{prec}(1)}^* \triangleq I$ and $N_{1|\text{prec}(1)}^* \triangleq N_1^* = I - \overline{J}_1^* J_1^*$ as shown in (3.10).

Proof. To proof that (3.32) fulfills (3.31) we first rewrite (3.32) as

$$J_i A^{-1} (S N_s)^T N_{i|\text{prec}(i)}^{*T} \prod_{(k>0) \wedge (k \neq i)}^N N_{k|\text{prec}(k)}^{*T}, \quad (3.34)$$

which can be done reorganizing null space terms according to the commutation property shown in (A.1). The first part of the above equation is equal to

$$J_i A^{-1} (S N_s)^T N_{i|\text{prec}(i)}^{*T} = J_{i|\text{prec}(i)}^* \Phi^* N_{i|\text{prec}(i)}^{*T} = 0. \quad (3.35)$$

Here we have used the equality $A^{-1} N_s^T = N_s A^{-1} N_s^T$ shown in (2.27), the equality $N_s = \overline{S N_s} S N_s$ shown in (2.44), the equality $\Phi^* N_{i|\text{prec}(i)}^{*T} = N_{i|\text{prec}(i)}^* \Phi^*$ demonstrated in (A.15), the property $(N_{i|\text{prec}(i)}^*)^2 = N_{i|\text{prec}(i)}^*$ demonstrated in (A.9), and the expression of $J_{i|\text{prec}(i)}^*$ given in (3.14). Using the expression of $N_{i|\text{prec}(i)}^*$ given in (3.33) and the expression of $\overline{J}_{i|\text{prec}(i)}^*$ given in (3.36), it is easy to demonstrate that the above term cancels out. \square

Corollary 3.2.4 (Dynamically consistent generalized inverse of $J_{k|\text{prec}(k)}^*$). *The following expression is a dynamically consistent generalized inverse of $J_{k|\text{prec}(k)}^*$*

$$\overline{J}_{k|\text{prec}(k)}^* \triangleq \Phi^* J_{k|\text{prec}(k)}^{*T} \left(J_{k|\text{prec}(k)}^* \Phi^* J_{k|\text{prec}(k)}^{*T} \right)^{-1}. \quad (3.36)$$

Proof. According to the dynamic projection shown in (3.16), the prioritized control structure shown in (3.20), and the prioritized inertia shown in (3.19) the following equality holds

$$J_k A^{-1} (S N_s)^T \Gamma_{k|\text{prec}(k)} = \left(\Lambda_{k|\text{prec}(k)} \right)^{-1} F_{k|\text{prec}(k)}, \quad (3.37)$$

where $\Gamma_{k|\text{prec}(k)}$ is a vector of prioritized torques and $F_{k|\text{prec}(k)}$ is a vector of prioritized control forces. Using (2.27) and (2.44), the expression of J_k^* shown in (3.5), and the property shown in (A.15) we can rewrite the above equation as

$$\Lambda_{k|\text{prec}(k)}^* J_k^* \Phi^* \Gamma_{k|\text{prec}(k)} = F_{k|\text{prec}(k)}. \quad (3.38)$$

Using the expression of $\Gamma_{k|\text{prec}(k)}$ given in (3.12), the property shown in (A.9), the expression of $J_{i|\text{prec}(i)}^*$ given in (3.14), and the expression of $\bar{J}_{k|\text{prec}(k)}^*$ given in (3.36), the above equation becomes

$$\bar{J}_{k|\text{prec}(k)}^{*T} \Gamma_{k|\text{prec}(k)} = F_{k|\text{prec}(k)}. \quad (3.39)$$

Null space torque components that integrate the term $I - \bar{J}_{k|\text{prec}(k)}^* J_{k|\text{prec}(k)}^*$ will map to zero control forces in the above equation. Because $\bar{J}_{k|\text{prec}(k)}^*$ is a generalized inverse of $J_{k|\text{prec}(k)}^*$ that cancels the effect of null space torques into task level forces and that provides the dynamic mapping shown in (3.39), we will refer to it as the dynamically consistent generalized inverse of $J_{k|\text{prec}(k)}^*$. \square

Corollary 3.2.5 (Compact expression of $N_{\text{prec}(k)}^*$). *The null space matrix given in (3.32) can be expressed using the following compact expression*

$$N_{\text{prec}(k)}^* = I - \sum_{i=1}^{k-1} \bar{J}_{i|\text{prec}(i)}^* J_{i|\text{prec}(i)}^*, \quad (3.40)$$

Proof. See Property A.0.4 in Appendix A. \square

3.2.4 Task Feasibility

Our motivation for task prioritization has been to filter out undesired effects on higher priority tasks due to lower priority tasks and to solve conflicting scenarios between task goals. In particular, to measure the feasibility of an arbitrary priority task k provided that all higher priority tasks are first accomplished, we propose to study the prioritized quantities $J_{k|\text{prec}(k)}^*$ given in (3.14) and $\Lambda_{k|\text{prec}(k)}^*$ given in (3.22). For instance, let us consider the stepping example shown in Figure 3.4. This joint locomotion and manipulation behavior

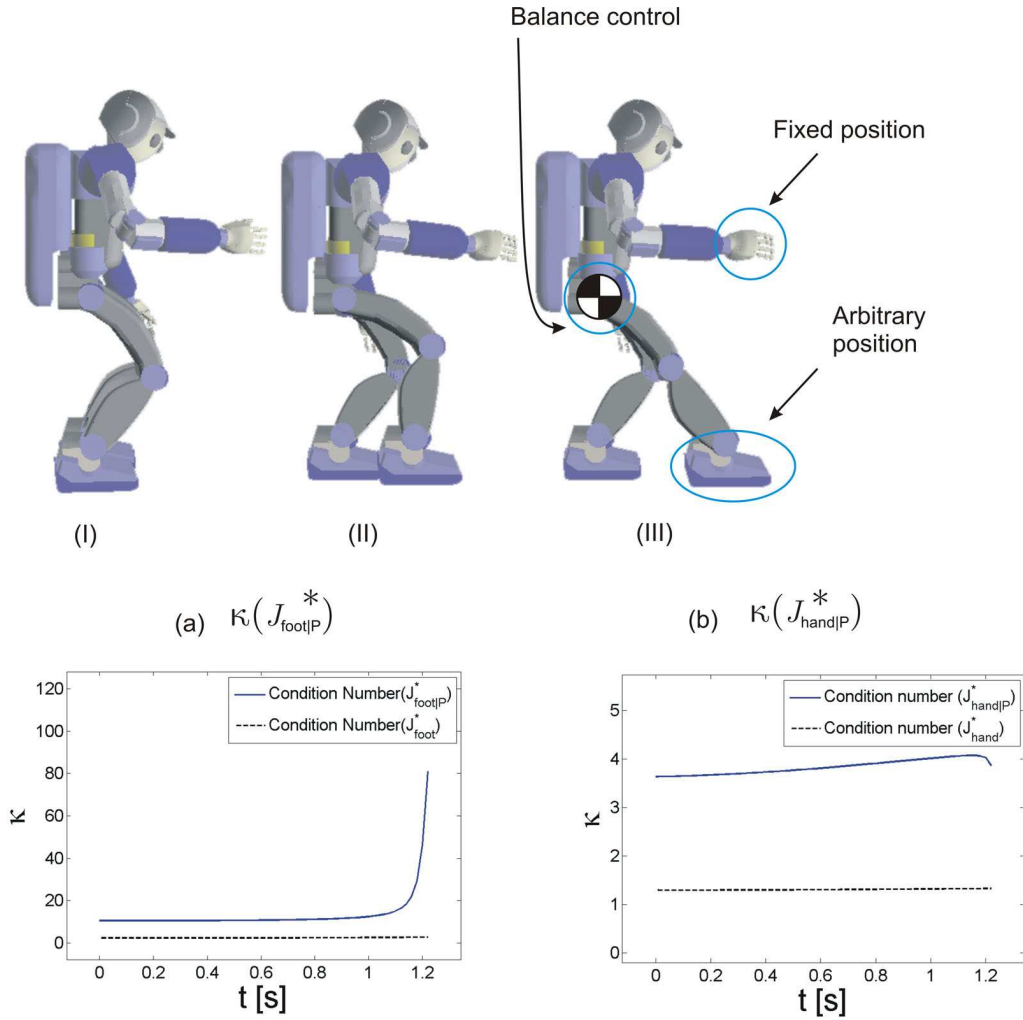


Figure 3.4: **Task feasibility under prioritization:** This snapshots from an actual experiment depict a joint manipulation and locomotion behavior generated using the task decomposition shown in Table 3.2. While the hand is teleoperated to a desired goal, the foot is commanded to step beyond its reachable workspace. To measure task feasibility, we monitor the condition numbers of the foot and hand task points under the task hierarchy shown in Table 3.3. While the condition number of the hand remains within normal values, the condition number of the foot grows towards infinity. This information allows us to change the behavior of the foot at runtime. In our example, when the condition number of the foot grows beyond normal values we command the foot to step back within the reachable workspace to gain stability. The symbols $J_{\text{hand}|P}^*$ and $J_{\text{foot}|P}^*$ indicate that the hand and foot tasks are subject to prioritized control.

has been created using the task decomposition shown in Table 3.2 with the task hierarchy shown in Table 3.3. This time the COG is commanded to remain on top of the left foot and the right hand is operated to remain at the initial location. The focus is on the right foot which is commanded to step forward beyond its reachable workspace. In practice such scenario could be the result of walking in rough terrain based on sensor information.

The data graphs accompanying Figure 3.4 depict the evolution of the prioritized Jacobians associated with the right foot and the right hand during movement execution. When the foot approaches the limit of the workspace, the condition number of the prioritized foot Jacobian grows beyond normal values. In contrast, the condition number associated with the prioritized hand Jacobian remains within normal values, indicating that there is enough movement redundancy to accomplish the desired hand behavior. Monitoring foot placement feasibility allows us to modify foot behavior during conflicting scenarios.

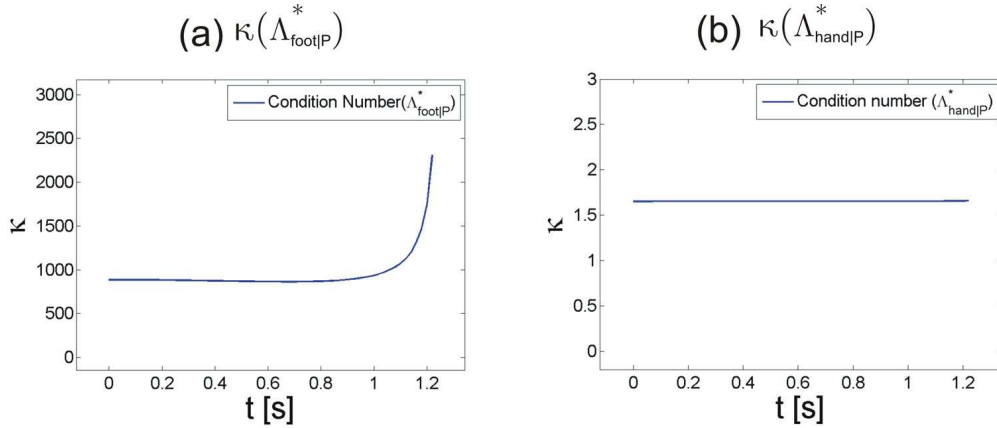


Figure 3.5: **Condition number of the prioritized inertias for Figure 3.4:** These graphs depict the evolution of the condition number of the prioritized inertias for the foot and hand tasks during movement execution. When the foot reaches full stretch the prioritized foot inertia becomes singular while the prioritized hand inertia stays within normal values.

Task feasibility can be measured by either evaluating the condition number of prioritized Jacobians, i.e.

$$\kappa(J_{k|\text{prec}(k)}^*) = \frac{\sigma_1(J_{k|\text{prec}(k)}^*)}{\sigma_{\dim(k)}(J_{k|\text{prec}(k)}^*)}, \quad (3.41)$$

where $\sigma_i(\cdot)$ corresponds to the i -th singular value of the enclosed term, or the condition

number of prioritized inertias, i.e.

$$\kappa(\Lambda_{k|\text{prec}(k)}^*) = \frac{\sigma_1(\Lambda_{k|\text{prec}(k)}^*)}{\sigma_{\dim(k)}(\Lambda_{k|\text{prec}(k)}^*)}. \quad (3.42)$$

For instance, in Figure 3.4 we depict the evolution of the condition number of the prioritized foot and hand Jacobians, abbreviated $\kappa(J_{\text{foot}|P}^*)$ and $\kappa(J_{\text{hand}|P}^*)$. Here the subscripts foot| P and hand| P are used to indicate that the foot and hand tasks are prioritized with respect to higher priority tasks as indicated in Table 3.3. As we can see, when the swinging leg approaches full stretch the prioritized Jacobian of the foot task becomes singular. However, the prioritized Jacobian of the hand task remains within normal values indicating that only the foot task is infeasible.

Data graphs showing the evolution of the condition numbers of prioritized inertias are shown in Figure 3.5. Similarly to the previous case, the foot prioritized inertia becomes singular while the hand prioritized inertia remains within normal values. Although either method can be used to measure task feasibility, an advantage of using prioritized inertias is that their dimension is much smaller compared to prioritized Jacobians. In fact, prioritized Jacobians have as many columns as the number of actuated joints.

3.2.5 Overview of Control Strategies

Operational space control yields linear control of task forces and accelerations, facilitating the implementation of a variety of control strategies involving position and force level behaviors. We will review now some acceleration-level control policies.

Goal-Based Position Control

During teleoperated or sensor-based position control, trajectories cannot be provided beforehand. Our approach is to move instantaneously in the direction of the commanded goals. This can be done by implementing a simple PD control law. However, to avoid reaching arbitrarily velocities and accelerations we need to integrate saturation strategies.

Velocity Saturation: This control method was proposed by (Khatib 1986) when presenting potential field techniques. It consists on implementing the following velocity-level controller

$$a_k^{ref} = -k_v(\dot{x}_k - \nu_v v_{des}), \quad (3.43)$$

$$v_{des} = \frac{k_p}{k_v}(x_k - x_{goal}), \quad \nu_v = \min\left(1, \frac{v_{max}}{\|v_{des}\|}\right), \quad (3.44)$$

where x_{goal} is the operational goal, ν_v is a velocity saturation factor, and v_{max} is the maximum allowable velocity. If $\|v_{\text{des}}\| < v_{\text{max}}$ this control law will be equivalent to a PD controller with the following behavior

$$\ddot{x}_k + k_v \dot{x}_k + k_p (x_k - x_{\text{goal}}) = 0. \quad (3.45)$$

If the desired velocity reaches values beyond the maximum allowable velocity, the above control law will yield the following velocity-based behavior

$$\ddot{x}_k + k_v \left(\dot{x}_k - v_{\text{max}} \frac{v_{\text{des}}}{\|v_{\text{des}}\|} \right) = 0, \quad (3.46)$$

which will maintain a constant velocity in the direction of the goal.

Simultaneous Acceleration and Velocity Saturation: We have designed a new PD controller that integrates saturation on both velocities and accelerations, and is expressed as follows

$$a_k^{ref} = \nu_a a_{\text{des}}, \quad (3.47)$$

$$a_{\text{des}} = -k_v (\dot{x}_k - \nu_v v_{\text{des}}), \quad \nu_a = \min\left(1, \frac{a_{\text{max}}}{\|a_{\text{des}}\|}\right), \quad (3.48)$$

$$v_{\text{des}} = \frac{k_p}{k_v} \nabla (x_k - x_{\text{goal}}), \quad \nu_v = \min\left(1, \frac{v_{\text{max}}}{\|v_{\text{des}}\|}\right), \quad (3.49)$$

where ν_a is an acceleration saturation factor, a_{max} is the maximum allowable acceleration, and v_{max} is the maximum allowable velocity. When $\|a_{\text{des}}\| < a_{\text{max}}$ this controller is equivalent to the previous velocity saturation controller. However when the desired accelerations are higher than the maximum allowable accelerations the resulting behavior is

$$\ddot{x}_k = a_{\text{max}} \frac{a_{\text{des}}}{\|a_{\text{des}}\|}, \quad (3.50)$$

which means that the task will accelerate in the direction of the desired goal with a value equal to a_{max} .

ZMP Control

Acceleration control can be used to control dynamic balance by manipulating the ZMP, the point on the ground where the horizontal moments due to ground reaction forces are equal to zero. To maintain balance stability the horizontal component of the ZMP, $x_{\text{zmp}H}$, needs

to be located within the stability polygon defined by the current placement of the feet. The control law that achieves this goal is (Vukobratovic and Borovac 2004; Sugihara 2004)

$$a_{\text{cogH}}^{\text{ref}} = \frac{f_{rz}}{M(x_{\text{cogz}} - x_{\text{zmpz}})}(x_{\text{cogH}} - x_{\text{zmpH}}^{\text{des}}), \quad (3.51)$$

where $x_{\text{zmpH}}^{\text{des}}$ is the desired location of the ZMP within the support polygon, f_{rz} is the vertical component of the reaction forces at the feet, M is the total mass of the robot, x_{cogz} is the current height of the robot's COG, x_{zmpz} is the ground level, and x_{cogH} is the horizontal position of the COG.

The ZMP horizontal position depends directly on COG accelerations according to the equation (Vukobratovic and Borovac 2004; Sugihara 2004)

$$x_{\text{zmpH}} = x_{\text{cogH}} - M\ddot{x}_{\text{cogH}} \frac{(x_{\text{cogz}} - x_{\text{zmpz}})}{f_{rz}}. \quad (3.52)$$

Therefore, we propose to apply the control law

$$\ddot{x}_{\text{cogH}} = a_{\text{cogH}}^{\text{ref}} \quad (3.53)$$

to achieve the desired behavior

$$x_{\text{zmpH}} = x_{\text{zmpH}}^{\text{des}}. \quad (3.54)$$

3.3 Examples

Let us study a few examples demonstrating the implementation of goal-oriented prioritized multi-task control. To implement the proposed control framework we have used a simulated humanoid called Collabot. Collabot is a robot measuring 1.60 *m* in height and weighting 83 *kg*. The kinematic model of the robot is similar to some of the existing high-end research humanoids. It has 29 DOF: 6 for each leg, 7 for each arm (3 shoulder, 1 elbow, 3 wrist), 1 for the chest yaw angle, and 2 for the head (pitch and yaw). The masses and inertias of the links are based on estimations from a real humanoid robot.

The examples we will show here were obtained using a simulation and control environment running on a PC at 2.13 GHz. Collabot's graphic model — provided by Honda Motor Co. — contains approximately 60,000 polygons. Our servo loop runs at 200 *MHz* on simulated time while the graphics are updated at 100frames/*s*. Collisions on the feet are computed using hierarchical distance models described in (Ruspini and Khatib 2000). The collision update rate is 10000*Hz*. A dynamic simulation environment based on (Chang and

Khatib 2000) takes the controller output and simulates the robot’s movement. Controller, simulator, and graphics, run at a speeds 4 times slower than real time for our 29 DOF robot.

3.3.1 Tracking the Contour of an Object

Our first example shown in Figure 3.6 involves tracking the contour of a large object (a sculpture) using hybrid force/position control. To create this behavior we have used the following task decomposition

Task Decomposition (Manipulation on object)

<i>Task Primitive</i>	<i>DOFs</i>	<i>Control Policy</i>	<i>Priority level</i>
Balance	2 ($x - y$ plane)	goal-based	1
Gaze orientation	2 (\perp plane)	goal-based	2
Right hand control	6 (y force, x moment)	hybrid force/position	3
Switching postures	$n = \text{number of joints}$	goal-based	4

Balance is achieved using velocity and acceleration saturation control as described in (3.47) and (3.48). The desired COG position is commanded to remain at the center of the supporting feet. The saturation values we use for the balance task are $v_{\max} = 1m/s$ and $a_{\max} = 3m/s^2$. These values are picked to avoid violating ZMP limits.

Gaze control is achieved by orienting the robot’s sight vector towards the teleoperated point (see Figure 3.7). The cartesian space error between the current head orientation and the desired goal orientation is

$$\phi_{\text{error}} = {}^{\phi}E_{\lambda} (\lambda_{\text{head}} - \lambda_{\text{goal}}), \quad (3.55)$$

where ${}^{\phi}E_{\lambda}$ is a 3×4 representation transformation between quaternions to cartesian space angles (Khatib 2004), λ_{head} is the quaternion representing the current head orientation, and λ_{goal} is the quaternion representing the desired orientation. Although ϕ_{error} is a 3×1 vector, only 2 DOFs are needed to orient the gaze towards the desired point. The orientation component around the head’s local z axis does not contribute to gaze orientation. Therefore when controlling the gaze task we do not need to use the full rotation Jacobian. Instead we remove the contributions from the z axis based on the following projection

$$J_{\text{gaze}} = ({}^0R_h S_{\text{gaze}}^T S_{\text{gaze}} {}^hR_0) J_{\text{head}(\phi)}. \quad (3.56)$$

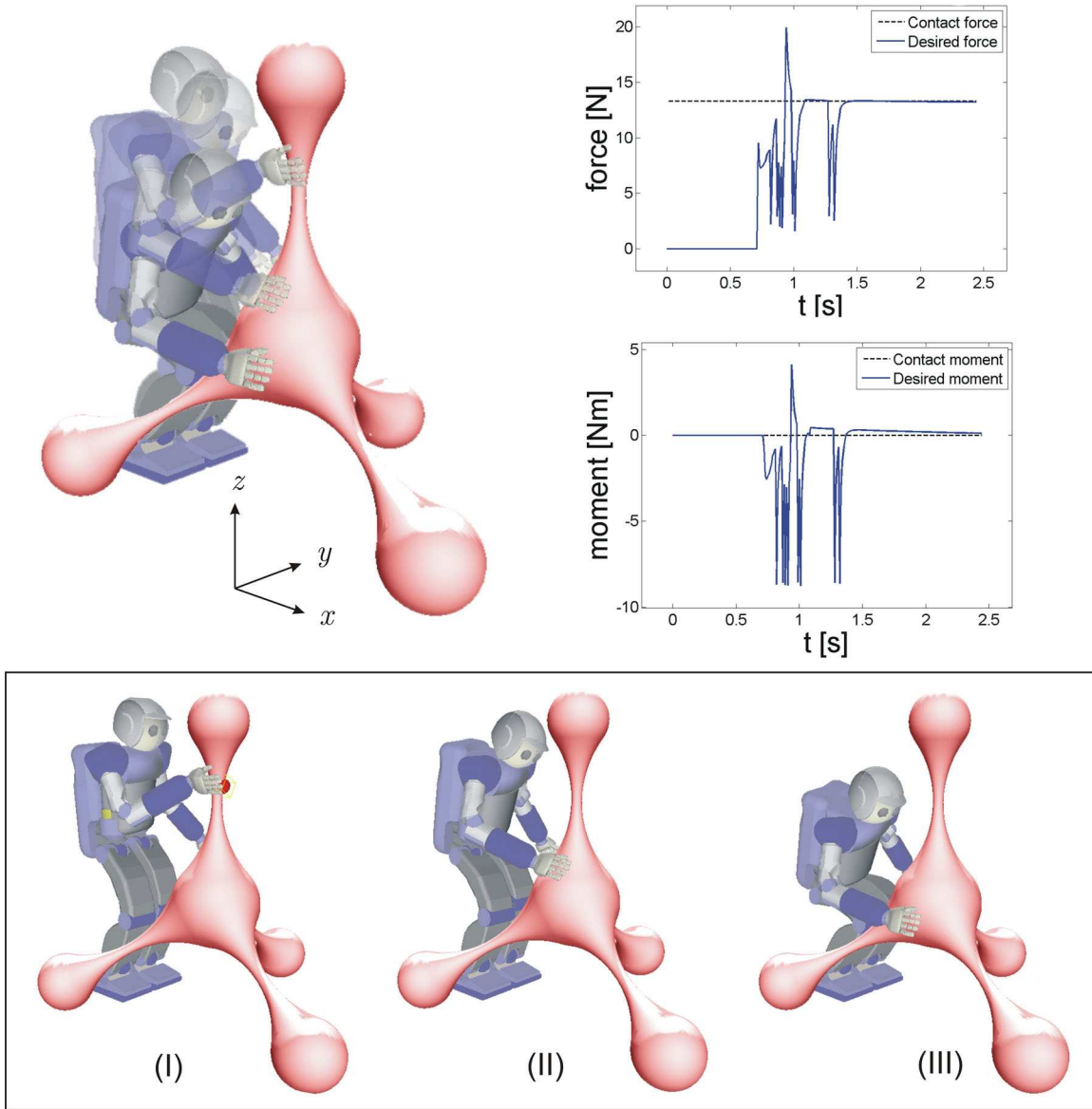


Figure 3.6: **Tracking the contour of an object:** This behavior has been created using the task decomposition shown in the previous table. Four tasks are simultaneously controlled: balance, gaze orientation, hybrid hand force/position, and posture switching. Linear forces perpendicular to the sculpture’s contour and moments tangent to the sculpture are simultaneously controlled to track the contour while maintaining a desired contact force against the surface.

Here $J_{\text{head}(\phi)}$ is the $3 \times (n + 6)$ Jacobian corresponding to head rotation coordinates, 0R_h is a rotation transformation from head coordinates to global coordinates, hR_0 is the transpose

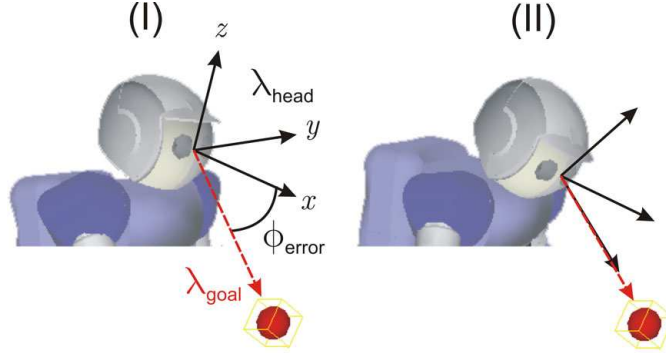


Figure 3.7: **Gaze control:** The head is controlled to track a teleoperated point.

of this transformation, and

$$S_{\text{gaze}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (3.57)$$

is a selection matrix that selects the x and y rotation DOFs. Therefore, the above transformation projects the head's rotation Jacobian into its local frame and removes the z component. The control policy used for gaze control is the velocity saturation law described in (3.43) adapted to the above rotation task, i.e.

$$a_{\text{ref(gaze)}} = -k_v \left(J_{\text{gaze}} \begin{pmatrix} \dot{\theta}_b \\ \dot{q} \end{pmatrix} - \nu_v \omega_{\text{des}} \right) \quad (3.58)$$

$$\omega_{\text{des}} = \frac{k_p}{k_v} \phi_{\text{error}}, \quad \nu_v = \min \left(1, \frac{\omega_{\text{max}}}{\| \omega_{\text{des}} \|} \right). \quad (3.59)$$

The robot's right hand is controlled using hybrid force/position control which will be discussed in a future paper in the context of prioritized control. Contact forces between the hand and the sculpture are controlled to push against the sculpture with a linear force on the y direction equal to $13N$ while the contact moments on the x axis are controlled for compliance. The cartesian position of the hand in the x and z directions is teleoperated while the rotation in the y and z directions is fixed to a value coplanar with the $y-z$ plane. During movement execution the hand is teleoperated to track up and down the contour of the sculpture. Contact is maintained at all times due to forces applied in the y direction. At the same time the hand rotates along the sculpture to maintain zero contact moments. The data graphs shown in Figure 3.6 show the evolution of the linear force on the y direction and the moment on the x direction in the first instants of contact.

The robot's postural stance is controlled using the methods we will describe in the

next chapter. We use the two snapshots shown in Figure 3.8 as desired poses. When the robot’s right hand is 35cm or higher above the ground, the robot’s posture is commanded to imitate the upright pose while when the hand reaches heights below 35cm the posture is commanded to imitate the crouching pose.

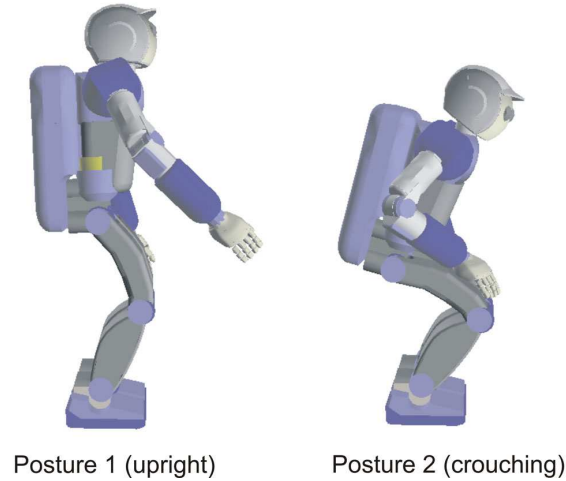


Figure 3.8: **Posture attractors:** Two static posture snapshots are used as posture attractors. For behaviors where the hand is above a predetermined height we command the posture to imitate the upright pose. When the hand goes below this threshold we command the robot to imitate the crouching pose.

3.3.2 Walking with Posture Variations

We consider the locomotion behavior shown in Figure 3.9. The robot’s feet are commanded to track desired goals while the posture is interactively controlled. In the first steps of Figure 3.9 the hip height is commanded to swing up and down, while in the final steps the upper-body orientation is commanded to swing back and forth. The walking behavior is created by sequencing a collection of whole-body movements (i.e. phases). Four phases are used to create the walking pattern as shown in Figure 3.10. In the first phase the robot’s COG is shifted towards the left foot. In the second phase, the right foot is commanded to step forward. The remaining two phases are counterparts of the previous two. This walking behavior is described as a collection of tasks, but no trajectories need to be provided beforehand. In fact, feet goals could be fed at runtime allowing the robot to step in different directions and with different walking patterns. Likewise, upper-body postural goals are changed at runtime. Changing the posture does not affect the stepping motion because our controller decouples tasks. In Figure 3.11 we plot the torque values obtained for the above

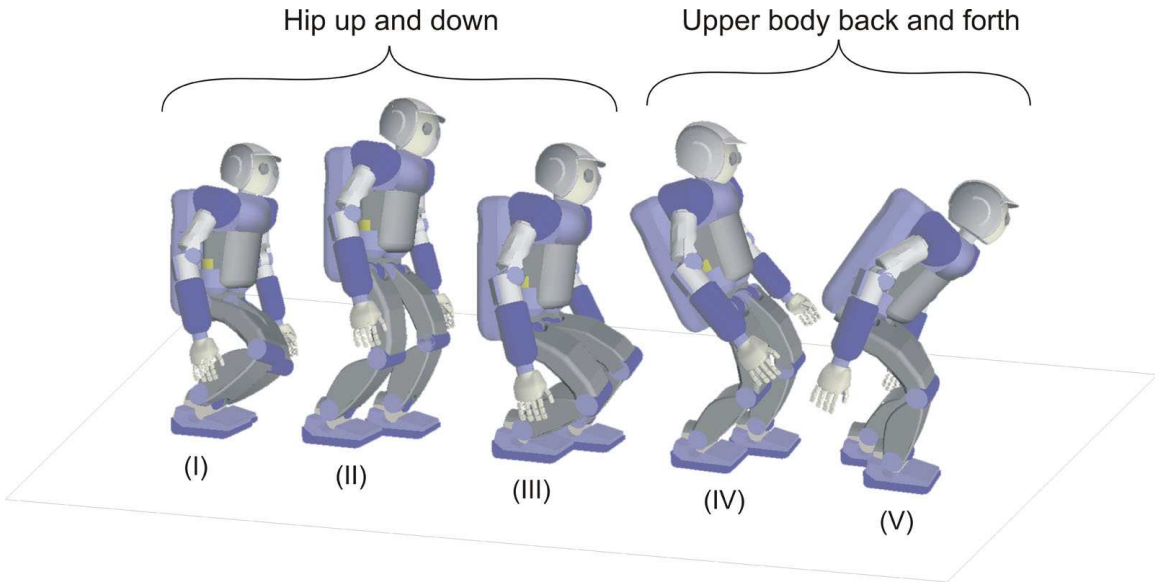


Figure 3.9: **Walking behavior with posture variations:** This behavior has been created using the walking sequence shown in Figure 3.10. Different phases are described in terms of task goals instead of relying on pre-programmed trajectories. Task goals can therefore be fed at runtime. In our example we command interactively the hip link to swing up and down and back and forth.

behavior. These values were recorded for an example with no hip variations.

3.3.3 Dynamic Walking: First Step

Dynamic balance is the technique used to create human-like walking and running behaviors in humanoid systems. It relies on COG accelerations instead of using COG positions only. COG accelerations are determined by the reaction forces acting on the supporting feet as well as by the action of gravity forces acting on the robot's COG. This balance of forces can be characterized by the following equation of motion

$$M(\ddot{x}_{\text{cog}} + G) = f_r, \quad (3.60)$$

where f_r is the 3×1 vector of reaction forces projected on the robot's COG (i.e. the sum of linear forces acting on supporting points), and G is the gravity acceleration vector, i.e.

$$G = \begin{pmatrix} 0 \\ 0 \\ -9.81m/s^2 \end{pmatrix}. \quad (3.61)$$

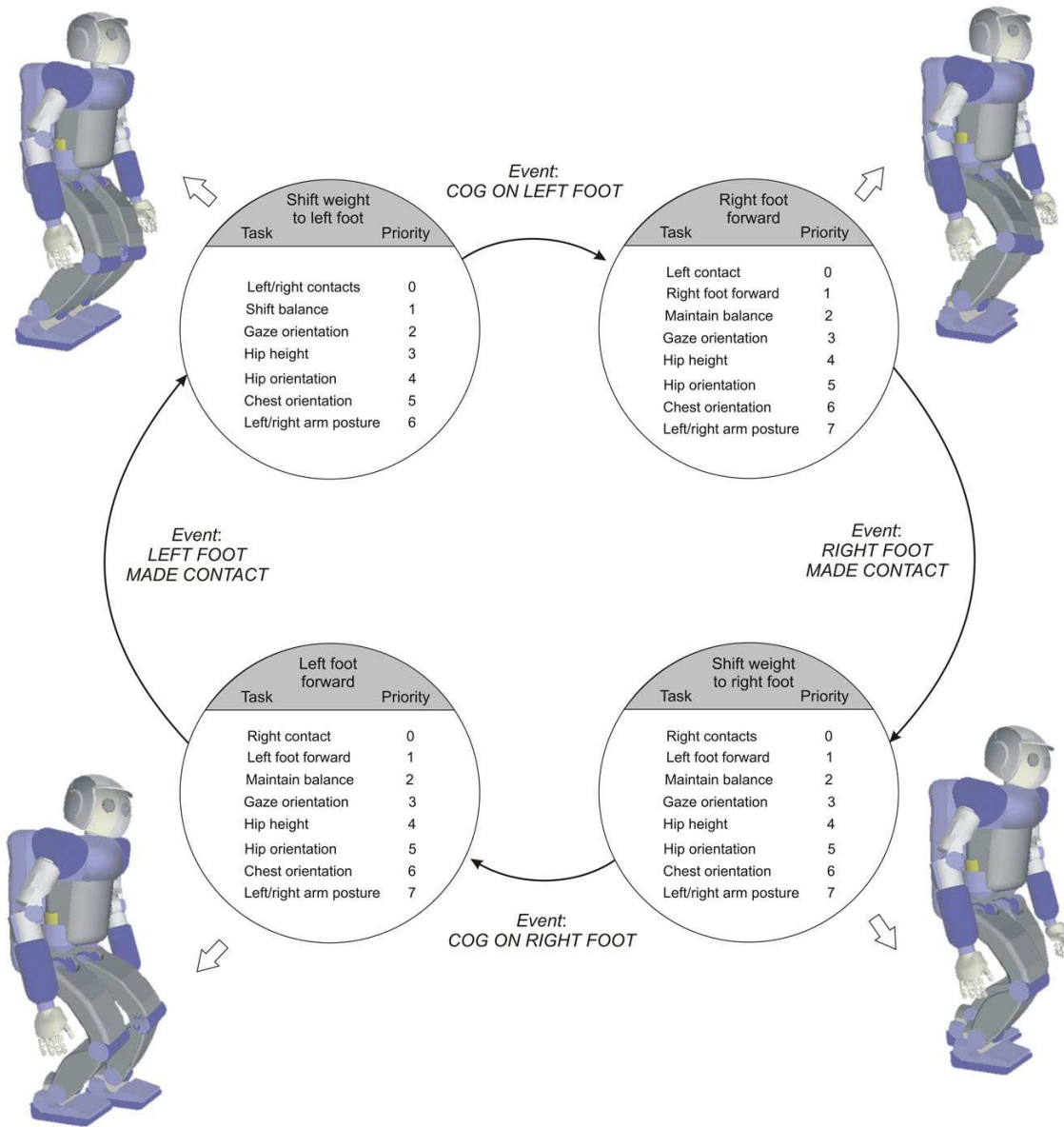


Figure 3.10: **Walking phases:** This diagram depicts the four walking phases and their decomposition into low-level tasks to accomplish the walking behavior of our experiment.

The ZMP (Vukobratovic and Borovac 2004), is used to characterize the robot’s ground stability during dynamic walking. The ZMP is the point on the ground where the horizontal moments due to ground reaction forces are equal to zero. In general, the moment of an arbitrary point x_p within the stability support polygon can be expressed as $m_p = x_p \times (-f_r)$.

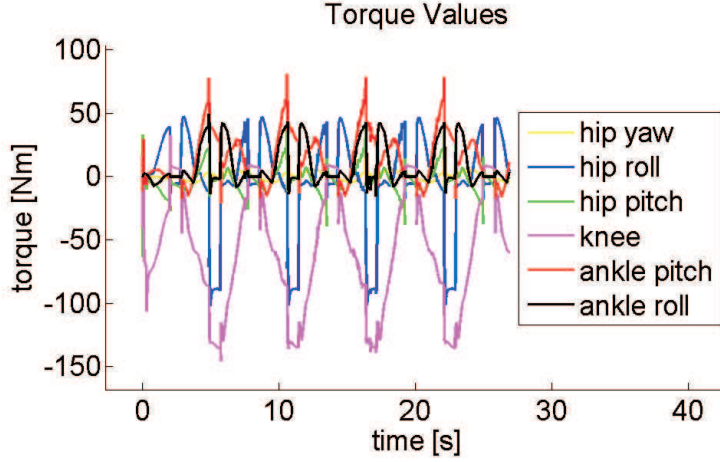


Figure 3.11: **Torques during waling motion:** These values correspond to torques on the right leg during the walking behavior described in the above example. No posture variations were used for this experiment, i.e. the hip link was kept at a fixed height and the torso orientation was kept upright at a fixed orientation. The highest values correspond to the torques on the knees during single support phases. The right hip’s roll joint (i.e. the hip abduction/adduction DOF) reached also high torque values because it sustains the lateral weight of the upper body when stepping forward.

Therefore the balance of moments between the COG and the ZMP can be formulated as

$$(x_{\text{cog}} - x_{\text{zmp}}) \times f_r + m_{\text{cog}} = m_{\text{zmp}}, \quad (3.62)$$

where m_{cog} and m_{zmp} are the moments around the COG and ZMP respectively, and \times is the vectorial product. To obtain the ZMP we set the horizontal moments to zero, i.e. $m_{\text{zmp}x} = m_{\text{zmp}y} = 0$. We also assume that m_{cog} is small and therefore can be ignored. Using the balance of forces of Equation (3.60), the previous equation yields the equality

$$x_{\text{zmp}H} = x_{\text{cog}H} - M \ddot{x}_{\text{cog}H} \frac{(x_{\text{cog}z} - x_{\text{zmp}z})}{f_{rz}}, \quad (3.63)$$

where $x_{\text{zmp}z}$ is the ground position where the ZMP is projected. This equality allows us to express COG accelerations as a function of the ZMP, i.e.

$$\ddot{x}_{\text{cog}H} = \frac{f_{rz}}{M(x_{\text{cog}z} - x_{\text{zmp}z})} (x_{\text{cog}H} - x_{\text{zmp}H}). \quad (3.64)$$

Although dynamic balance has been widely studied, our task oriented control approach allows us to control directly COG accelerations, facilitating the control of the ZMP. For

instance, let us consider the following control structure based on (3.20)

$$\Gamma = J_{\text{cogH}}^{*T} \left(\Lambda_{\text{cogH}}^* a_{\text{cogH}}^{\text{ref}} + \mu_{\text{cogH}}^* + p_{\text{cogH}}^* \right), \quad (3.65)$$

where we have assumed that the COG is controlled as the first priority task. When using the above control structure in (3.15) we obtain the linear behavior

$$\ddot{x}_{\text{cogH}} = a_{\text{cogH}}^{\text{ref}}. \quad (3.66)$$

Based on (3.64), the following reference vector provides control of dynamic balance

$$a_{\text{cogH}}^{\text{ref}} = \frac{f_{rz}}{M(x_{\text{cog}z} - x_{\text{zmp}z})} \left(x_{\text{cogH}} - x_{\text{zmpH}}^{\text{des}} \right), \quad (3.67)$$

where $x_{\text{zmpH}}^{\text{des}}$ is the desired ZMP location within the stability polygon. In fact, $x_{\text{zmpH}}^{\text{des}}$ determines the instantaneous direction and acceleration magnitude of the walk.

Because the generation of full dynamic walking patterns is an elaborate procedure (see Kajita et al. 2003a), for simplicity we will use the proposed ZMP controller to generate a single dynamic step. An example is illustrated in Figure 3.12 and has been created using the following task decomposition

Task Decomposition (Dynamic Walking)

<i>Task Primitive</i>	<i>DOFs</i>	<i>Control Policy</i>	<i>Priority level</i>
ZMP	2 ($x - y$)	COG accelerations	1
R foot position and orientation	6	position	2
Hip height	1	position	3
Hip orientation	3	position	4
Chest orientation	1	position	5
Arms posture	2×6	joint position	6

In Figure 3.12 we depict a forward step and a lateral step patterns. The accompanying data graphs correspond to the forward step. The ZMP is positioned within the supporting foot in a way that will accelerate the COG towards the landing position of the swinging foot. As a result the COG moves on a line within the stability polygon. To gain stability upon landing we command the COG to decelerate smoothly within the supporting polygon.

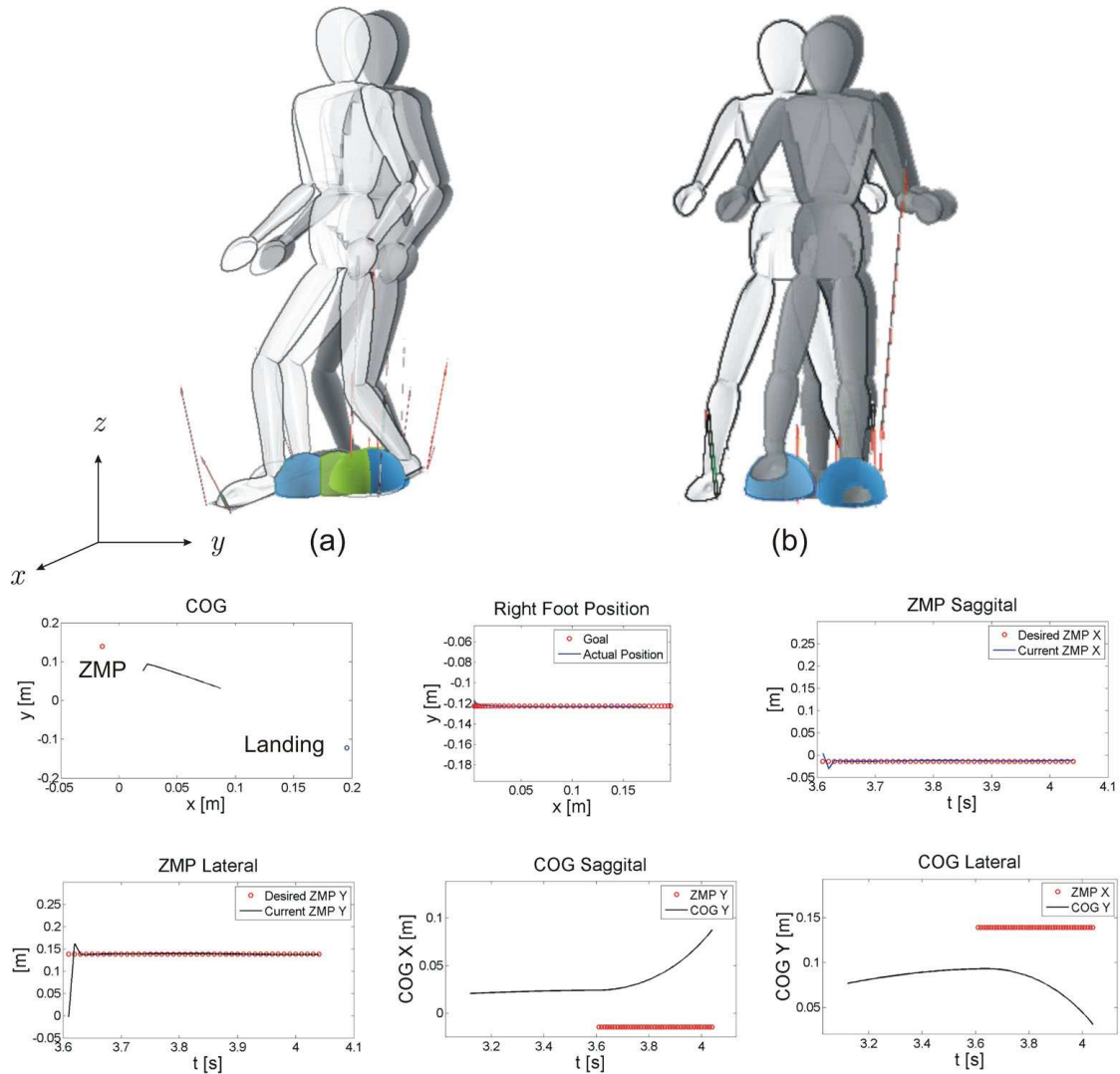


Figure 3.12: **Dynamic Walking (first step)** A single step is shown here using dynamic balancing. (a) depicts a forward step and (b) a lateral step. The data graphs correspond to the forward step. The blue sphere correspond to the ZMP and the green sphere to the horizontal projection of the COG. On the lateral step, only the ZMP sphere is shown.

Chapter 4

Posture Control

In this chapter we will develop methods for the control of postural tasks in humanoid systems. The coordination of the prioritized multi-task controller developed in the previous chapter and the posture controllers that will be developed in this chapter will provide the support to create interactive manipulation and locomotion behaviors while enhancing overall movement performance.

Posture control is an integral component of redundant manipulators and multi-legged robots where the goal is to enhance the execution of manipulation and locomotion behaviors. Earlier work on quadrupeds (Hirose et al. 1985; Raibert et al. 1986) discussed the role of the robot's posture to create efficient locomotion behaviors. More recently, postural behavior in living animals has been studied to inspire the control of robotic systems (Full 1993; Nelson and Quinn 1998). Postural behavior will be studied in detail in this chapter in the context of humanoid systems emphasizing interactive aspects for the realtime generation of whole-body behaviors.

Postural behavior plays an important role at different levels. For instance, the analysis of stiffness response in humans (Mussa-Ivaldi, Hogan, and Bizzi 1985) has inspired techniques to modulate postural response in robotic manipulators (Hogan 1987). More recently, new techniques to modulate postural dynamic behavior in biped and humanoid systems were developed at the actuator and control levels (Pratt, Dilworth, and Pratt 1997; Pratt 1995). Advanced redundant manipulators and whole-body control methods capable of modulating both the task's impedance and the null-space stiffness have been recently developed (Albu-Schaffer and Hirzinger 2002). In this chapter we will propose techniques to modulate postural stiffness in humanoid systems as well.

Postural behavior plays a key role in both implementing effective locomotion patterns

and optimizing manipulation performance. Liegeois addressed this issue by creating postural fields to keep joint angles close to pre-defined equilibrium positions (Liegeois 1977). Yoshikawa developed techniques to avoid kinematic singularities based on task manipulability optimization (Yoshikawa 1985). In this chapter we will address related performance criteria while bringing an additional requisite, imitating human poses. Imitating human poses is imposed in humanoid systems to facilitate their acceptance among humans. It is believed that human-like appearance in structure and movement plays an important role on the response of humans towards virtual characters and robots (Mori 2004). This hypothesis has been successfully exploited in computer animation and has become of increasing importance in robotics with the development of humanoid systems. Several android robots illustrating this concept have been recently developed (Ishiguro 2005). Our approach to address this problem will be to imitate human captured poses, a similar technique to the usage of key frames in film animation. However, in contrast with trajectory based techniques, our approach will not be based on interpolating between key frames. Instead, it will be based on imitating captured poses acting as attractors. In this context, captured poses will be used as equilibrium configurations projected into the task's residual redundancy, providing the support for arbitrary variations on task movement while imitating the desired poses. For instance, to enhance the reachable workspace during a manipulation task we will choose captured poses that are away from joint limits. Multiple capture poses supported by a switching policy will be considered to address large variations on the task.

From an efficiency perspective, actuation effort plays a key role in postural behavior. Using musculoskeletal models of the human body and a dynamic simulation environment we recently observed a strong correlation between gravity torques and muscle's torque capacity (Khatib, Warren, Desapio, and Sentis 2004). Based on this observation we conjectured that human postural behavior involves minimization of muscle effort and defined a mathematical representation of effort in the form of a Euclidean norm of gravity torques weighted by muscle torque capacities. In this chapter we will implement effort minimization in posture space based on this effort potential. A similar effort potential based on unweighed torques was proposed and implemented by (Boulic and Mas 1995) in the context of inverse kinematic control.

This chapter is organized as follows. In Section 4.1 we will characterize the task's residual redundancy for posture control extending the prioritized controller presented in the previous chapter. In Section 4.2 we will develop task based control methods for controlling the robot's posture. In Section 4.2 we will develop gradient descent methods to optimize the robot's postural behavior based on optimal criteria. We will discuss the imitation of

human captured poses and the minimization of actuation effort. Finally, in Section 4.4 we will discuss the modulation of postural stiffness to provide safe whole-body contact interactions.

4.1 Posture Control Structure

In the previous chapter, we developed a prioritized multi-task controller characterized by the control structures shown in (3.11) and (3.30), i.e.

$$\Gamma = \sum_{k=1}^N \Gamma_{k|\text{prec}(k)}, \quad \Gamma_{k|\text{prec}(k)} = J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)}. \quad (4.1)$$

Similarly to the whole-body torque decomposition shown in (2.76), when multiple prioritized tasks are simultaneously controlled, the residual redundancy defines the posture space of motion and can be characterized by the following structure:

Theorem 4.1.1 (Whole-body prioritized control structure). *The following control structure provides linear control of N prioritized tasks and defines the postural space of motion*

$$\Gamma = \sum_{k=1}^N J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)} + N_t^{*T} \Gamma_{\text{posture}}, \quad (4.2)$$

where,

$$N_t^* \triangleq I - \sum_{k=1}^N \bar{J}_{k|\text{prec}(k)}^* J_{k|\text{prec}(k)}^*, \quad (4.3)$$

is the null-space of all prioritized tasks and can be derived using the recursive expression presented in (3.40). Here, N_t^* signifies the null space of all priority tasks.

Proof. While the terms $J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)}$ provide linear control of task forces and accelerations as discussed in Theorem 3.2.1, N_t^* acts as a base for the control of residual DOFs as discussed in Corollary 3.2.3, and therefore Γ_{posture} defines the control of the robot's postural motion with no coupling effects on priority tasks. \square

In general, we will use the following more compact notation equivalent to (4.2)

$$\Gamma = \sum_{k=1}^N \Gamma_{k|\text{prec}(k)} + \Gamma_{p|t} = \Gamma_t + \Gamma_{p|t}, \quad (4.4)$$

where

$$\Gamma_{p|t} \triangleq N_t^{*T} \Gamma_{\text{posture}} \quad (4.5)$$

is the vector of prioritized postural torques. Here the subscript $p|t$ is used to indicate that postural torques are projected in the null space of all priority tasks.

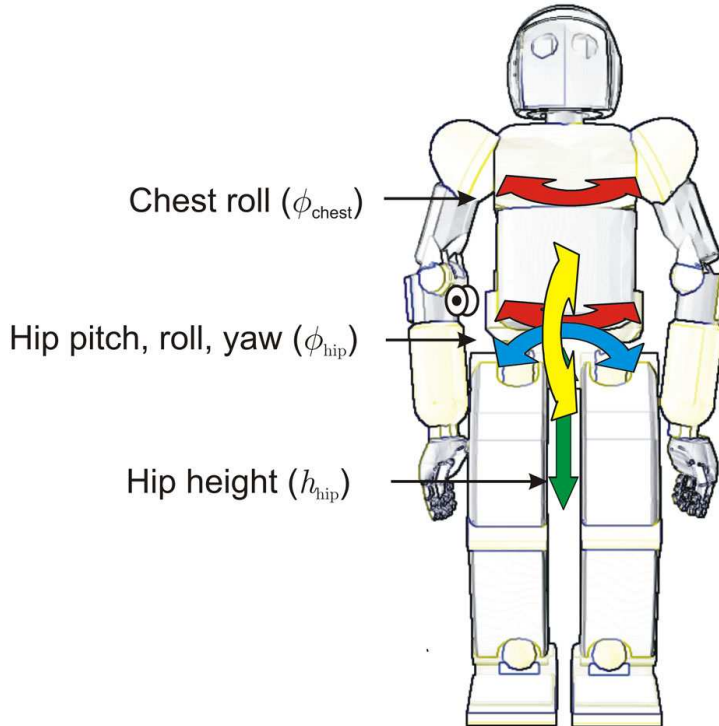


Figure 4.1: **Postural DOFs in the main body:** Posture movement associated with the main body is due to hip attitude and height and chest vertical orientation. By directly controlling these DOFs we can control the robot's posture.

We consider two different methods to control the robot's posture. The first consists on controlling pre-determined task points on the robot's body associated with postural motion (see Figure 4.1). The second consists on optimizing desired postural criteria in joint space via gradient projection. Let us start describing the first method.

4.2 Task Based Postures

The first approach to controlling the task's residual redundancy is an extension of multi-task control of the previous chapter. If in the previous chapter we addressed multi-task

control as the control of multiple operational points used to maintain balance stability and to interact with the environment, we now consider the control of additional task points describing the robot's internal motion. For instance, let us consider a humanoid equipped with 6 DOFs per leg, 1 DOF in the upper body (e.g. vertical orientation), 3 DOFs in the head, and 7 DOFs per arm. Considering the robot is standing up, postural DOFs on the main body involve the 3D orientation and the vertical position of the hip's link and the vertical orientation of the chest's link (see Figure 4.1).

Controlling these degrees of freedom is independent of manipulation, locomotion, and balance tasks. Therefore, each postural DOF can be individually controlled towards desired goals. For instance, to keep the body upright we can directly command the hip's upright orientation to align with the direction of the gravity field. Similarly, to keep the hip at a certain height from the ground we can command the hip's vertical DOF to reach the desired position.

Let us consider the following coordinate representation for the different postural DOFs

$$x_{\text{posture}} \triangleq \begin{pmatrix} h_{\text{hip}} \\ \lambda_{\text{hip}} \\ \phi_{\text{chest}} \end{pmatrix} \in \mathcal{R}^6, \quad (4.6)$$

where h_{hip} represents the hip's link vertical position, λ_{hip} represents the hip's link orientation (using Euler parameters), and ϕ_{chest} represents the chest's vertical orientation as shown in Figure 4.1. We can simultaneously control all postural DOFs by handling x_{posture} as a macro task and associating the following joint Jacobian

$$\dot{x}_{\text{posture}} = J_{\text{posture}} \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad J_{\text{posture}} = \begin{pmatrix} J_{h,\text{hip}} \\ J_{\lambda,\text{hip}} \\ J_{\phi,\text{chest}} \end{pmatrix}, \quad (4.7)$$

where $J_{h,\text{hip}}$, $J_{\lambda,\text{hip}}$, and $J_{\phi,\text{chest}}$ are the Jacobians associated with the different postural DOFs and can be obtained from the basic Jacobian representation of the hip and chest links as shown in (3.3).

Let us now consider the posture's dynamic behavior by left-multiplying (2.24) by the

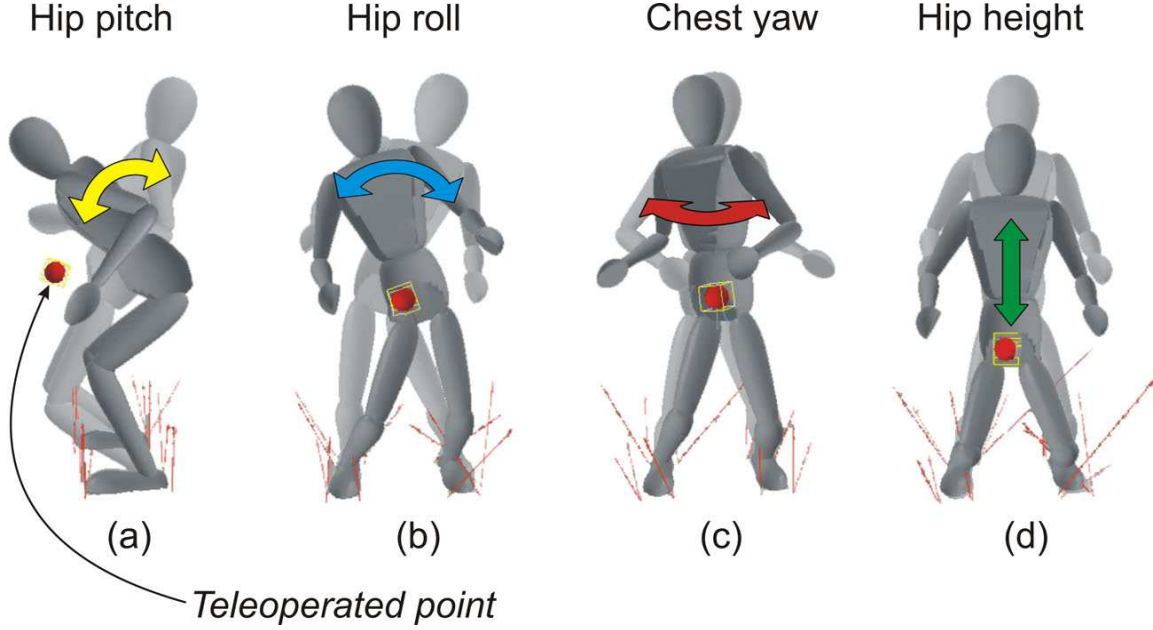


Figure 4.2: **Goal-based posture control:** These snapshots from an actual experiment are obtained using goal-based control of postural DOFs. The COG's horizontal component is controlled to remain at the center of the feet to maintain balance stability, while arms and head are controlled in joint space to remain fixed.

term $J_p A^{-1}$, leading to the following equation of motion

$$\ddot{x}_p - \dot{J}_p \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + J_p A^{-1} N_s^T (b + g) + J_p A^{-1} J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = J_p A^{-1} (S N_s)^T \left(\Gamma_{p|t} + \sum_{k=1}^N \Gamma_{k|\text{prec}(k)} \right). \quad (4.8)$$

Notice that we use the abbreviated notation x_p and J_p to represent the postural coordinates and the associated Jacobian shown in (4.6) and (4.7). For the above representation we have used the equality

$$\ddot{x}_p = J_p \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} + \dot{J}_p \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} \quad (4.9)$$

and the torque decomposition shown in (4.4).

Based on the prioritized multi-task control structures presented in (3.20) and (3.26), we propose to control postural DOFs using the following posture control vector:

Proposition 4.2.1 (Posture motion control). *The following torque vector yields linear control of postural accelerations*

$$\Gamma_{p|t} = J_{p|t}^{*T} \left(\Lambda_{p|t}^* a_p^{ref} + \mu_{p|t}^* + p_{p|t}^* - \Lambda_{p|t}^* J_p A^{-1} (S N_s)^T \sum_{k=1}^N \Gamma_{k|prec(k)} \right), \quad (4.10)$$

where a_p^{ref} is a joint feedback control policy for all postural DOFs, $J_{p|t}^* \triangleq J_p^* N_t^*$ is the prioritized posture Jacobian discussed in (3.14),

$$\Lambda_{p|t}^* = \left(J_{p|t}^* \Phi^* J_{p|t}^{*T} \right)^{-1}, \quad (4.11)$$

is a posture inertial term similar to (3.19), and $\mu_{p|t}^*$ and $p_{p|t}^*$ are Coriolis/centrifugal and gravity terms with similar expressions than (3.27) and (3.28).

Proof. Because this controller is based on Corollary 3.2.1, it will yield the linear behavior

$$\ddot{x}_{posture} = a_p^{ref}. \quad (4.12)$$

□

Choosing the following aggregation of control policies

$$a_p^{ref} = \begin{pmatrix} a_{h,hip}^{ref} \\ \alpha_{\lambda,hip}^{ref} \\ \alpha_{\phi,chest}^{ref} \end{pmatrix}, \quad (4.13)$$

where $a_{h,hip}^{ref}$, $\alpha_{\lambda,hip}^{ref}$, and $\alpha_{\phi,chest}^{ref}$ are control policies for the different postural DOFs, will result in linear control of postural coordinates, i.e.

$$\ddot{x}_{posture} = a_p^{ref} \iff \begin{cases} \ddot{h}_{hip} = a_{h,hip}^{ref} \\ \ddot{\lambda}_{hip} = \alpha_{\lambda,hip}^{ref} \\ \ddot{\phi}_{chest} = \alpha_{\phi,chest}^{ref} \end{cases}. \quad (4.14)$$

For instance, for the example shown in Figure 4.2, we have used the following joint PD

control law

$$a_p^{ref} = -K_p \begin{pmatrix} h_{\text{hip}} - h_{\text{goal}}(t) \\ \lambda_{\text{hip}} - \lambda_{\text{goal}}(t) \\ \phi_{\text{chest}} - \phi_{\text{goal}}(t) \end{pmatrix} - K_v \begin{pmatrix} \dot{h}_{\text{hip}} \\ \dot{\lambda}_{\text{hip}} \\ \dot{\phi}_{\text{chest}} \end{pmatrix}. \quad (4.15)$$

where K_p and K_v are desired proportional and differential gain matrices. The goal positions are determined using a 6-D haptic device connected to the proxy red sphere shown in Figure 4.2.

4.2.1 Example: Posture Control to Avoid an Overhead Obstacle

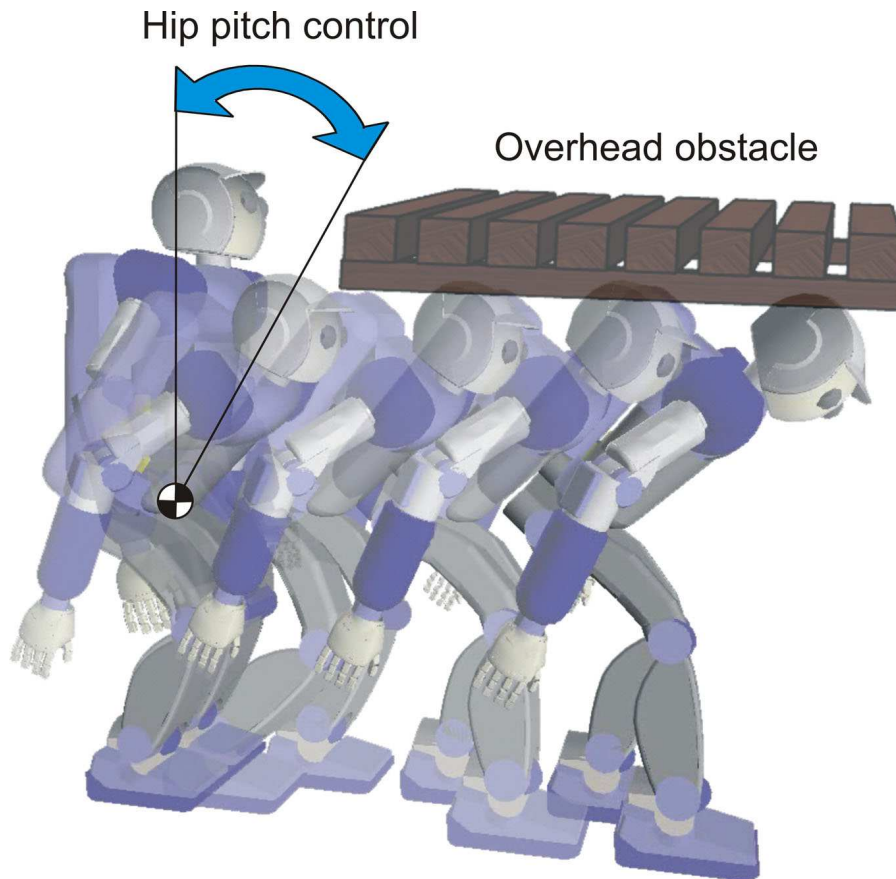


Figure 4.3: **Posture control for avoiding overhead obstacle:** This sequence of snapshots of an actual experiment illustrates the interactive control of upper-body orientations during walking. The walking behavior is created using the primitives described in 3.10, while the hip's link pitch orientation is commanded to rotate forward 45° from the vertical axis.

An example implementing realtime interactive control of the posture during a walking behavior is shown in Figure 4.3. Here, walking is created using the four phases described in Figure 3.10. The posture is controlled based on the techniques previously shown in this section. To avoid colliding with the overhead obstacle, the hip's pitch orientation is commanded to rotate 45° forward with respect to the global vertical axis. Because the posture is controlled in the null space of tasks involved in the walking behavior, all goals can be simultaneously accomplished at runtime.

4.3 Criteria Based Postures

A second approach to controlling the robot's posture consists on optimizing desired criteria through gradient projection in the task's null space. Although null space control of secondary criteria has been widely used mostly in the context of inverse kinematic control, we will propose here a novel projection technique that will provide optimal gradient descent at the torque level. Using this technique we will implement novel postural control strategies that will mimic human postural behavior and optimize actuator performance in humanoids.

In the previous section we presented techniques to control the position and orientation of postural DOFs. However, it is often more convenient to determine desired postural criteria, and optimize it using all available movement redundancy. Our representation of humanoids as non-fixed branching systems will allow us to characterize the task's residual redundancy at the whole-body level and use it to optimize the desired criteria.

Supported by anthropomorphic bodies, humanoids are designed to transcend conventional machines. While prioritized multi task control provided a platform to implement interactive locomotion and manipulation behaviors, the objective of the robot's posture is to optimize performance criteria and mimic human behavior. The posture control strategies we will discuss in this section will be designed to achieve such performance.

4.3.1 Imitation of Human Poses

Our goal here is to develop control strategies to mimic captured human poses (see Figure 4.4). In contrast with other approaches, our control strategy will not rely on interpolating trajectories between poses. Instead we will use poses as attractor potentials.

Let us consider the two captured poses shown in Figure 4.4. One of the snapshots corresponds to an upright pose while the other one corresponds to a crouching pose. The musculoskeletal data used in the capturing process has been derived from SIMM models

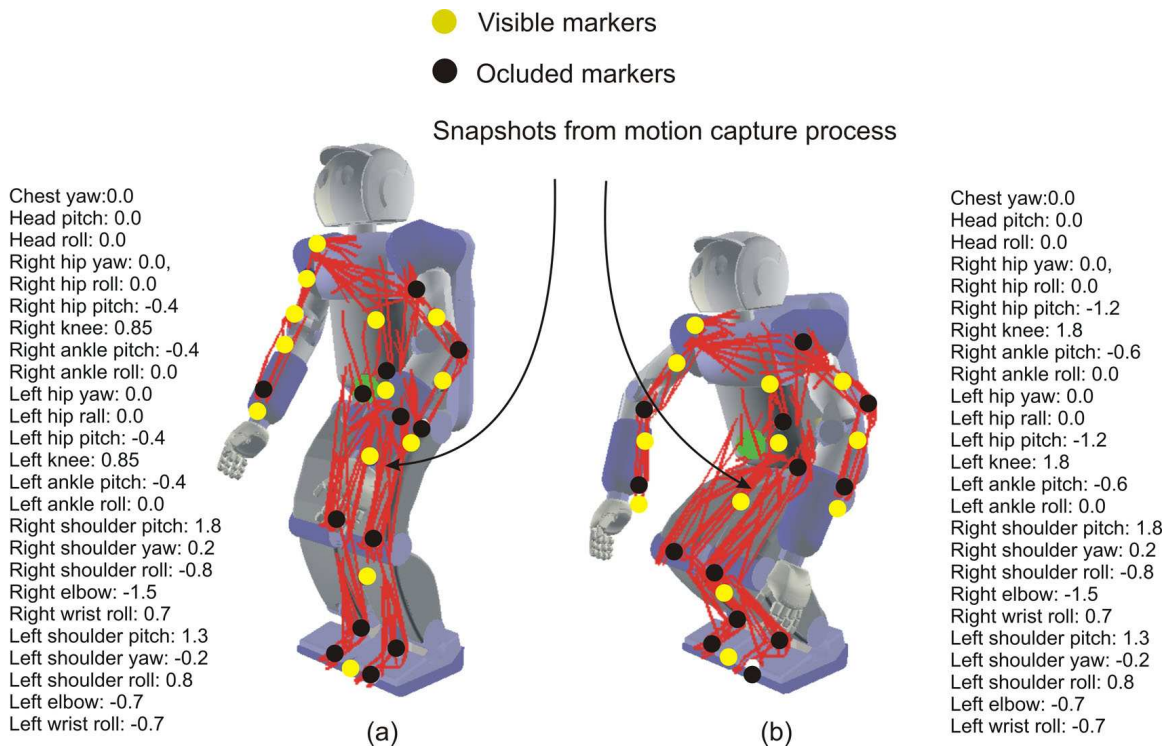


Figure 4.4: **Captured human poses:** Concept depicting the mapping of captured human poses into robot poses. The above images connected with virtual muscles (red lines) are the outcome of a motion capture process of a real human standing up and crouching down. These poses are mapped into our humanoid model to be used as equilibrium poses by our posture controller. The tags adjacent to the superimposed images correspond to joint positions of the captured poses.

(Delp and Loan 2000). Our hypothesis is that human-like movement will emerge when imitating locally captured human poses. To avoid interfering with the robot's global task, the desired poses will be used as attractor potentials operating in the null space of operational tasks.

Let us consider the control of the robot's posture to mimic a single pose attractor (e.g. the upright pose in the previous example). Captured poses are represented in terms of joint positions by the vector q_{pose} .

Definition 4.3.1 (Postural criteria). *The following potential function defines an error function in joint space designed to imitate recorded poses*

$$V_p(q_s) = \| q_p - q_{\text{pose}} \|^2. \quad (4.16)$$

Here q_p is the vector of joint coordinates for posture control and $\| \cdot \|^2$ is the Euclidean norm.

Definition 4.3.2 (Joint coordinates for posture control). *The following vector represents a subset of joint positions assigned to control postural criteria*

$$q_p \subset q. \quad (4.17)$$

In turn, postural velocities are expressed as

$$\dot{q}_p = S_p \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (4.18)$$

where S_p is a predefined posture selection matrix that acts like a Jacobian.

Definition 4.3.3 (Prioritized postural Jacobian). *The following matrix represents the prioritized postural Jacobian with similar derivation than (3.14)*

$$S_{p|t}^* = S_p^* N_t^*, \quad (4.19)$$

where N_t^* is the null space matrix shown in (4.3) and

$$S_p^* = S_p \overline{SN_s} \quad (4.20)$$

is the constrained postural Jacobian with similar derivation than (2.47).

Dynamics and Control

We first derive the equation of motion of postural coordinates by left-multiplying (3.15) by the term $S_p A^{-1}$, leading to the following equation of motion

$$\ddot{q}_p + S_p A^{-1} N_s^T (b + g) + S_p A^{-1} J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = S_p A^{-1} (SN_s)^T (\Gamma_{p|t} + \Gamma_t), \quad (4.21)$$

where we have used the equality

$$\ddot{q}_p = S_p \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} \quad (4.22)$$

and the torque decomposition given in (4.4). We seek control solutions that yield linear control of postural accelerations while operating in the task's null-space as shown in (4.5).

Definition 4.3.4 (Prioritized inverse postural inertia). *The following expression is referred to as the prioritized inverse posture inertia*

$$\Phi_{p|t}^* \triangleq S_{p|t}^* \Phi^* S_{p|t}^{*T}, \quad (4.23)$$

where Φ^* is the constrained projection of A^{-1} shown in (2.42). In general the above inverse inertia will not be full rank because q_p is normally larger than the number of available DOFs within the tasks's residual redundancy, i.e.

$$\text{rank}(S_p) > \text{rank}(N_t^*). \quad (4.24)$$

Although $\Phi_{p|t}^*$ does not have a strict physical meaning it will appear when formulating control structures as an inertial term.

Theorem 4.3.1 (Postural criteria-based control). *The following posture control vector will yield optimal gradient descent of postural criteria*

$$\Gamma_{p|t} = S_{p|t}^{*T} F_{p|t}, \quad (4.25)$$

where $F_{p|t}$ is a posture control vector with the following expression

$$F_{p|t} = \left(\Phi_{p|t}^* \right)^+ \alpha_p^{ref} + b_{p|t}^* + g_{p|t}^* - \left(\Phi_{p|t}^* \right)^+ S_p A^{-1} (S N_s)^T \Gamma_t. \quad (4.26)$$

Here $(.)^+$ is the Moore-Penrose pseudo-inverse, α_p^{ref} is a control policy implementing gradient descent of postural criteria, and the following vectors are Coriolis/centrifugal and gravity terms

$$b_{p|t}^* \triangleq \left(\Phi_{p|t}^* \right)^+ S_p A^{-1} N_s^T b + \left(\Phi_{p|t}^* \right)^+ S_p A^{-1} J_s^T \Lambda_s J_s \begin{pmatrix} \dot{\vartheta}_b \\ \dot{q} \end{pmatrix}, \quad (4.27)$$

$$g_{p|t}^* \triangleq \left(\Phi_{p|t}^* \right)^+ S_p A^{-1} N_s^T g. \quad (4.28)$$

Proof. Based on the above control terms, (4.21) becomes

$$\ddot{q}_p + S_p A^{-1} N_s^T (b + g) + S_p A^{-1} J_s^T \Lambda_s \dot{J}_s \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} = \Phi_{p|t}^* F_{p|t} + S_p A^{-1} (S N_s)^T \Gamma_t, \quad (4.29)$$

where we have used the following equality shown in Property (4.3.1)

$$\Phi_{p|t}^* = S_p A^{-1} (S N_s)^T S_{p|t}^{*T}. \quad (4.30)$$

Because $\Phi_{p|t}^*$ is normally not full rank, we consider its eigen-decomposition

$$\Phi_{p|t}^* = \begin{pmatrix} U_r & U_n \end{pmatrix} \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U_r^T \\ U_n^T \end{pmatrix}, \quad (4.31)$$

where the U matrices are eigenvectors and Σ_r is the matrix of non-zero eigenvalues. Here, the zero eigenvalues correspond to uncontrollable posture directions. The number of zero eigenvalues is equal to the number of operational coordinates being controlled. Therefore, the unitary matrix U_r (the subscripts r and n mean rank and null respectively) reveals the controllable directions where the posture operates. Based on the above eigen-decomposition the pseudo-inverse of the posture inertia can be expressed as

$$\left(\Phi_{p|t}^* \right)^+ = U_r \Sigma_r^{-1} U_r^T. \quad (4.32)$$

Plugging this expression into (4.26) and using (4.25) into (4.29) we obtain the linear behavior

$$U_r^T \left(\ddot{q}_p = \alpha_p^{ref} \right), \quad (4.33)$$

revealing that our controller linearizes the controllable posture directions while ignoring other directions dominated by priority tasks. To obtain the above equation we have used the equality $\Phi_{p|t}^* \left(\Phi_{p|t}^* \right)^+ = U_r U_r^T$ and the equality $U_r^T U_r = I$. In turn, plugging α_p^{ref} will result in optimal gradient descent. \square

Property 4.3.1 (Alternative expression of $\Phi_{p|t}^*$). *The inverse posture inertia defined in (4.23) has the alternative expression given in (4.30).*

Proof. Using the expression of $S_{p|t}^*$ given in (4.19), the property $N_t^* \Phi^* = \Phi^* N_t^{*T}$ shown in (A.15), and the property $(N_t^*)^2 = N_t^*$ shown in (A.10) the following equality holds

$$S_{p|t}^* \Phi^* S_{p|t}^{*T} = S_p^* \Phi^* S_{p|t}^{*T}. \quad (4.34)$$

Using the expression of S_p^* given in (4.20) and the expression of Φ^* given in (2.42) the above expression becomes

$$S_p \overline{SN_s} SN_s A^{-1} (SN_s)^T S_{p|t}^{*T}. \quad (4.35)$$

Using the equality $\overline{SN_s} SN_s = N_s$ given in (2.44), the equality $N_s A^{-1} = A^{-1} N_s^T$ given in (2.27), and the property $(N_s)^2 = N_s$ given in (2.26) the above expression becomes

$$S_p A^{-1} (SN_s)^T S_{p|t}^{*T} \quad (4.36)$$

which is equal to (4.30). \square

Proposition 4.3.1 (Gradient descent control law). *The following PD control law implements postural gradient descent while providing a velocity saturation mechanism*

$$\alpha_p^{ref} = -k_v (\dot{q}_p - \nu_v \omega_{des}) \quad (4.37)$$

$$\omega_{des} = -\frac{k_p}{k_v} \nabla V_p, \quad \nu_v = \min\left(1, \frac{\omega_{max}}{\|\omega_{des}\|}\right). \quad (4.38)$$

where ν_v is the saturation term, ω_{max} is the maximum allowable angular velocity, ∇V_p is the gradient of V_p with respect to q_p , and k_p and k_v are proportional and differential gains respectively.

Proof. When applying this control law to (4.33), the posture will descend along the controllable directions defined by $U_r^T \nabla V(q_p)$, achieving optimal descent within the projection hyper-plane defined by the directions U_r . At the same time, this control law will result in velocity saturation in the controllable directions according to the description given in (3.43). \square

4.3.2 Example 1: Upright Posture Attractor

Let us consider the control example shown in Figure 4.5 designed to mimic an upright posture. The only acting task besides the posture is balance control. Based on the task and posture control decomposition described in (4.4) we consider the following dual control structure

$$\Gamma = \Gamma_{\text{cogH}} + \Gamma_{p|t}, \quad (4.39)$$

where Γ_{cogH} is the torque term designed to control the horizontal COG as shown in Section 2.4 and $\Gamma_{p|t}$ is the posture torque structure. The upright pose shown in Figure 4.5 corresponds to the desired pose reference q_{pose} discussed in (4.16). In our example we use all

joint coordinates to define the pose, therefore $q_p = q$. To minimize the posture potential we use the control law described in (4.37). The focus is now on the posture. The robot is

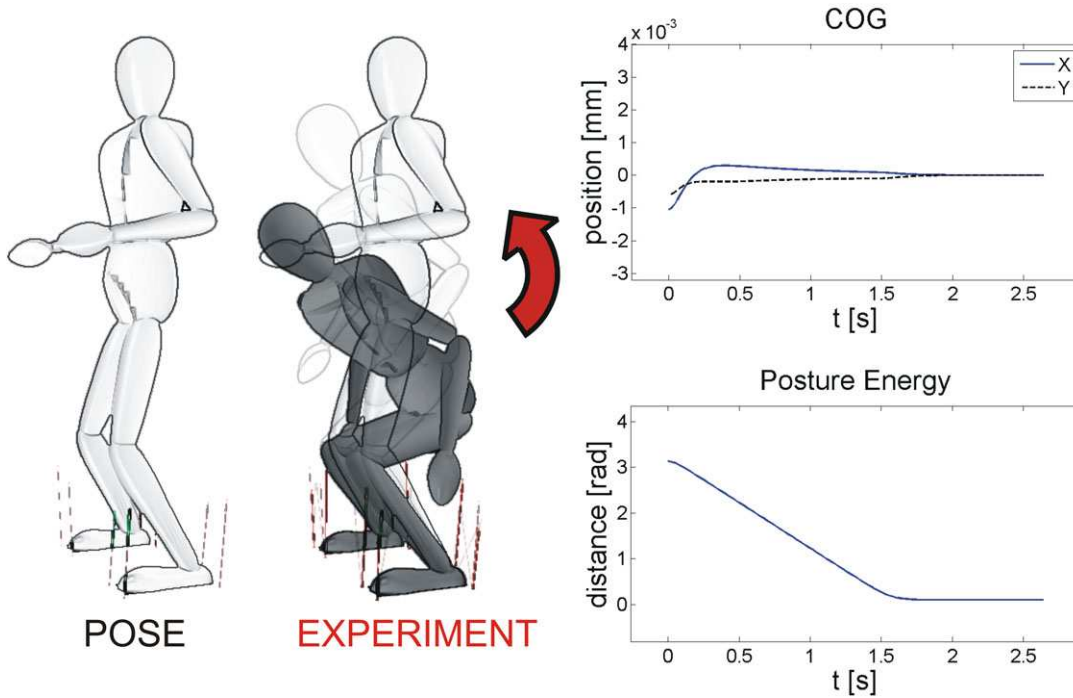


Figure 4.5: **Upright pose attractor:** In this experiment, the joint space distance to a captured pose is minimized under gradient projection in posture space. The robot is initially set on a crouched pose. Upon movement activation the robot’s body moves upwards to minimize the posture potential. The data graphs depict the evolution of the posture potential towards the captured pose and the evolution of the robot’s horizontal COG.

initially set on a crouched pose. Upon activating the posture controller, the body moves upwards towards the reference pose. The posture energy is minimized within the limits allowed by balance constraints. The minimum energy achieved is $0.1rad$, a value very close to zero, since the capture pose was balanced in first place. Using task and posture decomposition we are able to maintain balance while minimizing the posture energy. The robot’s horizontal COG is maintained at the center of the feet at all times despite postural motion. The maximum error on the COG is less than $0.5mm$, demonstrating that the task is completely decoupled from the posture. Posture velocity saturation can be observed in the accompanying data graphs. A value $\omega_{max} = 2rad/s$ has been used to saturate the angular velocity. After reaching maximum velocity the energy decreases at a fixed rate.

4.3.3 Example 2: Upright Posture Control with Contact Task

We consider the example shown in Figure 4.6 where the robot's right hand is commanded to maintain contact with the pallet's jack while optimizing the robot's pose. The robot's hand is controlled to stay at a fixed location, making contact with the pallet jack. The whole-body controller has the structure

$$\Gamma = \Gamma_{\text{cogH}} + \Gamma_{\text{hand|cogH}} + \Gamma_{p|t}, \quad (4.40)$$

where $\Gamma_{\text{hand|cogH}}$ corresponds to hand control acting with lower priority than balance control. The starting pose has been chosen to be off-symmetry, with the upper torso leaning to the

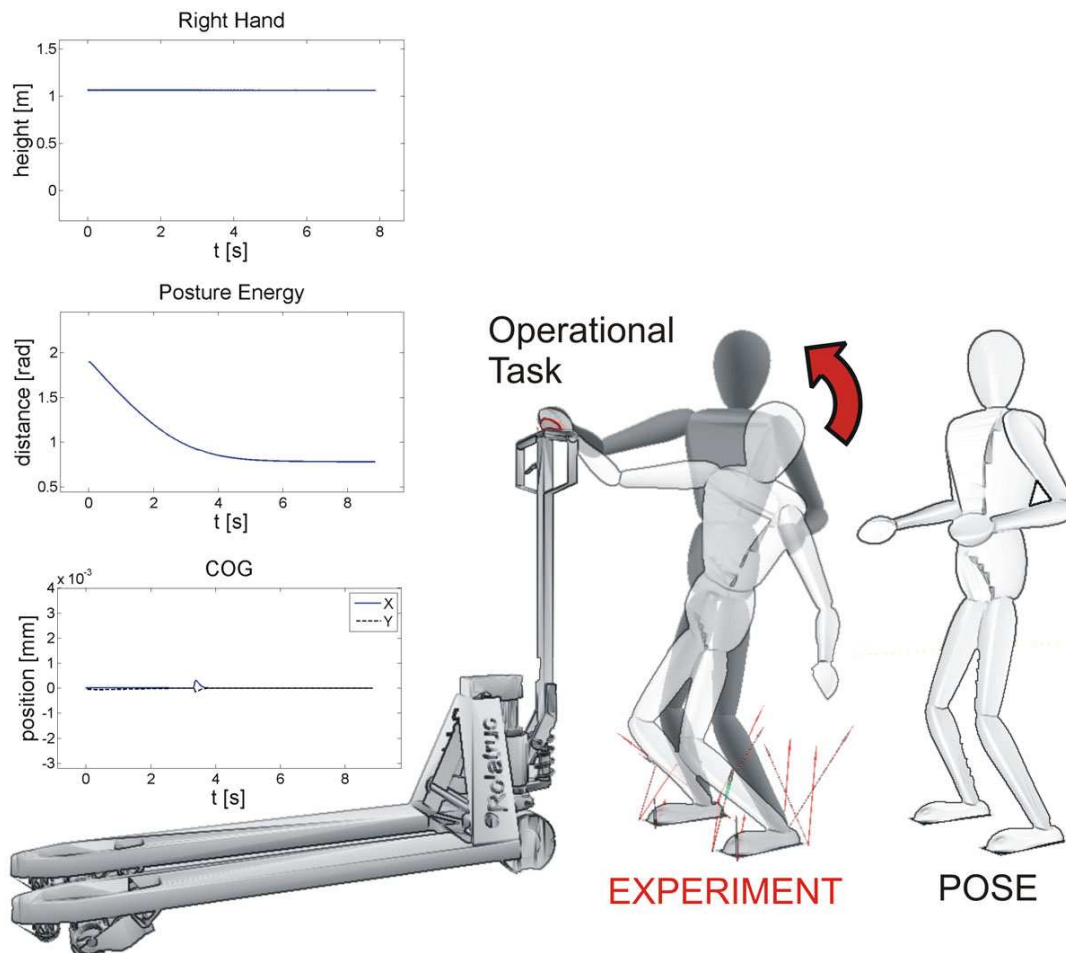


Figure 4.6: **Upright pose under contact:** Here, the robot's right hand is controlled to maintain contact with the pallet jack while the posture imitates the upright reference pose.

left. Upon posture activation the robot's body moves upwards to minimize the posture potential. Although the starting posture was non-symmetrical, the final posture inherits the symmetry of the reference pose. In the data graphs, we can see that both the hand position and the COG's horizontal position remain quasi static during movement execution demonstrating the decoupled effect between tasks and posture.

4.3.4 Example 3: Posture Behavior with Interactive Task

We consider the example shown Figure 4.7) based on the same control structure as in (4.40). This time however, the right hand is teleoperated in realtime to reach different points around the reachable workspace. The hand moves over a large area, approaching the ground at $t = 5s$ and reaching the highest position at $t = 12s$. The posture potential is optimized at all times minimizing the distance to the reference pose.

4.3.5 Example 4: Posture Switching To Optimize Reachable Workspace

We consider the example shown in Figure 4.8 where two poses are used as references and a switching policy between poses is implemented. Using multiple poses allows us to use more efficiently the reachable workspace while maintaining higher resemblance to human poses. In Figure 4.8 two different experiments are shown side by side, one involving posture switching and another one based on a single pose. The two reference poses correspond to the upright pose and the crouching pose shown in Figure 4.4. The task consists on reaching the ball. While the experiment shown on the right side of Figure 4.8 is implemented using a single reference pose, the experiment on the left side is implemented using postural switching. This last experiment is supported by the following posture potentials,

$$V_{\text{pose 1}} = \| q - q_{\text{upright}} \|^2, \quad (4.41)$$

$$V_{\text{pose 2}} = \| q - q_{\text{crouching}} \|^2, \quad (4.42)$$

where q_{upright} and $q_{\text{crouching}}$ correspond to the reference poses. The controller structure is

$$\Gamma = \Gamma_{\text{cogH}} + \Gamma_{\text{hands|cogH}} + \Gamma_{p|t}. \quad (4.43)$$

The right and left hands are simultaneously controlled. The switching policy establishes that the upright pose is used for reaching tasks below a given height and the crouching pose is used otherwise. Initially the robot is standing up. The goal is to reach the ball at ground level. When reaching down, the posture energy decreases until the pose resembles

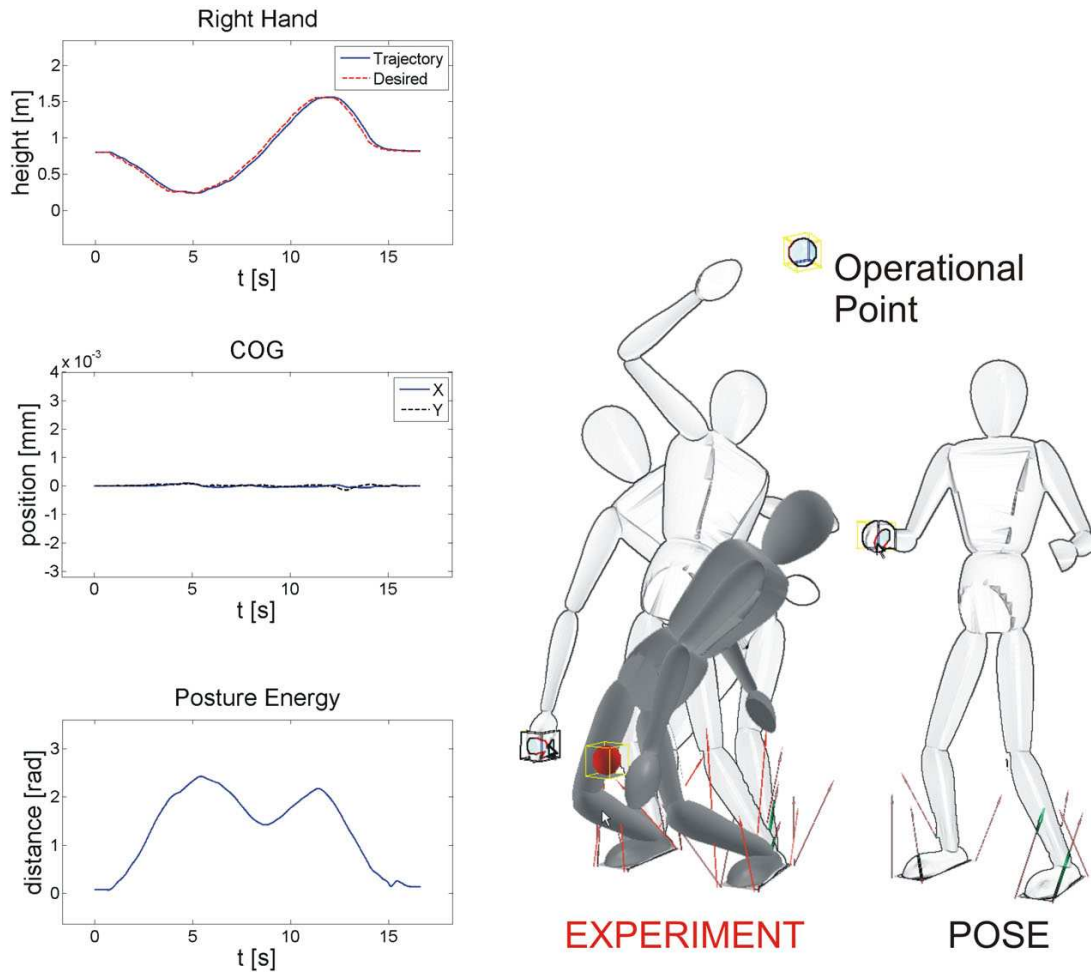


Figure 4.7: **Posture behavior with interactive task:** These images correspond to a task and posture control example where the robot’s right hand is teleoperated to different points around the workspace and the posture is used to minimize the distance an upright pose.

the upright reference, however as the movement proceeds downwards the posture energy increases again. At $t = 4s$ the switching threshold is reached and as a result the crouching pose is activated. Once more the posture energy decreases in its effort to imitate the new pose.

On the right side, a similar experiment is shown based only on imitating the upright pose, i.e. no switching policy between postures is implemented. As a result the posture energy increases steadily departing from the reference pose.

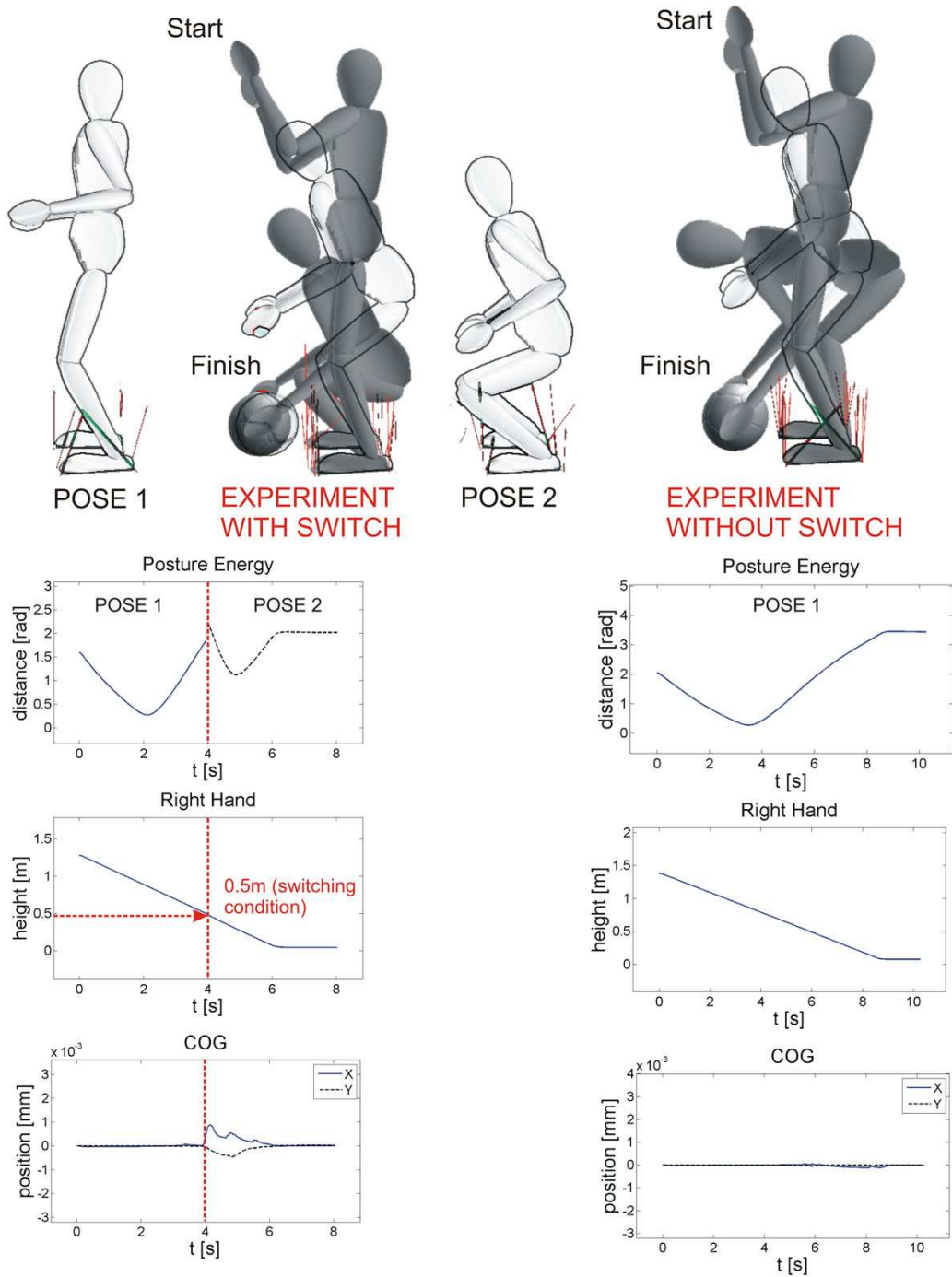


Figure 4.8: **Posture switching:** In this example, two reference poses are used for posture control. Pose 1 is used for upper-body reaching tasks at heights above 0.5m, and pose 2 is used for lower-body reaching tasks.

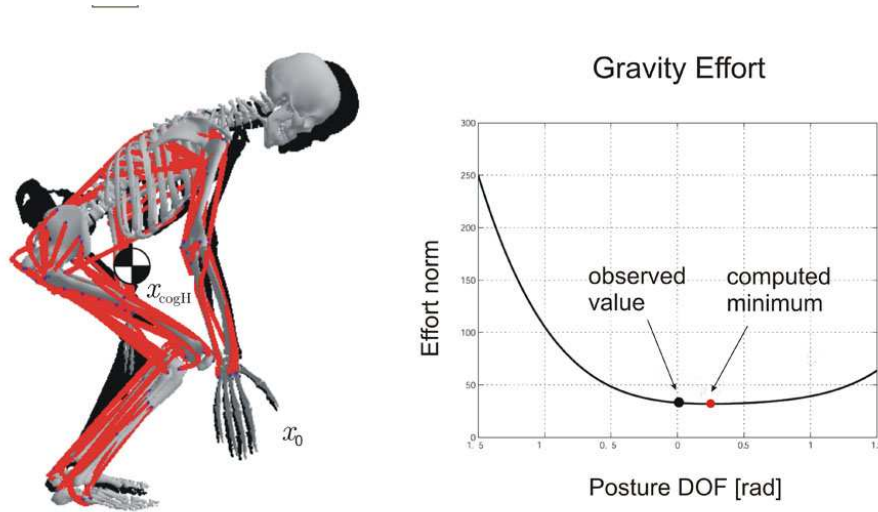


Figure 4.9: **Human effort:** Plot showing human gravity effort compared against computed null-space motion in posture space using the posture potential given in (4.44). The observed final configuration, is 0.27 radians (15.5°) from the computed minimum.

4.3.6 Effort Minimization

In a recent paper we demonstrated that during reaching tasks, humans distribute gravity torques over the various joints in a manner that is correlated with the available torque capacity at each joint (Khatib, Warren, Desapio, and Sentis 2004). Based on this observation, and past work from the biomechanics community (Anderson and Pandy 2001; Crowninshield and Brand 1981), we conjectured that postural motion involves the minimization of a muscle effort potential, $V_p(q)$, with the following expression,

$$V_p(q) = \sum_{k=1}^n w_i \frac{g_i(q)^2}{\Gamma_{B_i}(q)^2}, \quad (4.44)$$

where g_i is the gravity torque acting on joint i , Γ_{B_i} is the muscle induced boundary torque (upper or lower boundary depending on the sign of g_i), and w_i is an arbitrary weighting term. We validated this potential in a simulated musculoskeletal model of the human body suggesting a robust correlation between the recorded motion and the proposed postural potential. In Figure 4.9 we show a captured human pose with the observed effort valued compared against the computed minimum using the potential function given in (4.44). As we can see, the final configuration of the real and simulated human are very close in value (0.27 radians), validating the proposed effort potential.

Based on these observations we have implemented gravity effort minimization for humanoid posture control using similar control representations than in previous examples. In fact, we also use the control structure described in 4.25 to minimize effort. However, the reference acceleration vector α_p^{ref} is now determined based on an effort energy. We first characterize the vector of gravity torques as

$$g(q) = S g(x_b, q) = \begin{pmatrix} g_6(x_b, q) \\ \vdots \\ g_{n+6}(x_b, q) \end{pmatrix}, \quad (4.45)$$

where S is the actuation matrix given in (2.19) and $g(b, q)$ is the vector of generalized torques described in (2.4). This selection matrix removes components corresponding to passive DOFs. Furthermore, we use the following simplified effort potential

$$V_p(g(q)) = \|g(q)\|^2 = \sum_{k=1}^n g_i(q)^2. \quad (4.46)$$

To optimize the above potential we propose the following velocity saturated control law

$$\alpha_p^{ref} = -k_v(\dot{q} - \nu_v \omega_{des}) \quad (4.47)$$

$$\omega_{des} = -\frac{k_p}{k_v} \nabla_q V_p(g(q)), \quad \nu_v = \min\left(1, \frac{\omega_{max}}{\|\omega_{des}\|}\right), \quad (4.48)$$

where $\nabla_q V_p(g(q))$ is the gradient of the effort energy with respect to joint coordinates. The gradient of the above effort potential can be further expressed using the following decomposition

$$\nabla_q V_p(g(q)) = \frac{\delta V_p(q)}{\delta q} = \frac{\delta V_p(g(q))}{\delta g(q)} \frac{\delta g(q)}{\delta q} = J_{g(q)}^T g(q), \quad (4.49)$$

where $J_{g(q)} = \partial g(q)/\partial q$ is the Jacobian of the gravity torque vector. To determine the gravity Jacobian we use the following numerical approximation

$$J_{g(q)} = \begin{pmatrix} \frac{g_1(q_1+\Delta q_1)-g_1(q_1)}{\Delta q_1} & \dots & \frac{g_1(q_n+\Delta q_n)-g_1(q_n)}{\Delta q_n} \\ \vdots & & \vdots \\ \frac{g_n(q_1+\Delta q_1)-g_n(q_1)}{\Delta q_1} & \dots & \frac{g_n(q_n+\Delta q_n)-g_n(q_n)}{\Delta q_n} \end{pmatrix}. \quad (4.50)$$

An alternative closed form expression of the above Jacobian can be found in (Baerlocher

2001) and has the following expression

$$J_{g(q)} = \left(\frac{\partial g(q)}{\partial q_1} \quad \dots \quad \frac{\partial g(q)}{\partial q_n} \right), \quad (4.51)$$

with

$$\frac{\partial g(q)}{\partial q_i} = J_{\text{cog}}(G \times a_j). \quad (4.52)$$

Here, J_{cog} is the Jacobian of the global COG as expressed in (2.52), $G = \begin{pmatrix} 0 & 0 & -9.81m/s^2 \end{pmatrix}^T$ is the gravity acceleration vector, and a_j is the 3×1 unit axis of rotation (e.g. for a revolute joint rotating on its y axis, $a_j = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T$).

4.3.7 Example 1: Effort Minimization while Standing Up

We consider the example shown in Figure 4.10 illustrating the proposed effort minimization method acting on our simulated humanoid. The simultaneous task and posture control structure is expressed as

$$\Gamma = \Gamma_{\text{cogH}} + \Gamma_{p|t}, \quad (4.53)$$

where Γ_{cogH} is the torque that controls the horizontal components of the COG and $\Gamma_{p|t}$ is the torque that controls the proposed effort posture according to the structures previously discussed.

The humanoid is initially set on a crouched pose. Upon posture activation, the gradient of the effort potential described in (4.46) is minimized using the control law (4.47) and based on the control structure developed in (4.25). The following control parameters are used in our example: $\omega_{\text{max}} = 2rad/s$, $k_p = 1000$, and $k_v = 2\sqrt{k_p}$.

In the data graph accompanying Figure 4.10 we can observe that the posture energy descends at a moderate speed and reaches values close to zero.

4.3.8 Example 2: Effort Minimization with Interactive Task

We consider the example shown in Figure 4.11. This time, two operational tasks are considered consisting on controlling balance as well as the robot's right hand. The torque decomposition is characterized by the structure

$$\Gamma = \Gamma_{\text{cogH}} + \Gamma_{\text{head|cogH}} + \Gamma_{\text{hand|head|cogH}} + \Gamma_{p|t}, \quad (4.54)$$

where $\Gamma_{\text{head|cogH}}$ corresponds to the a head orientation task subject to the balance task and $\Gamma_{\text{hand|head|cogH}}$ corresponds to hand control subject to balance and head control. Posture

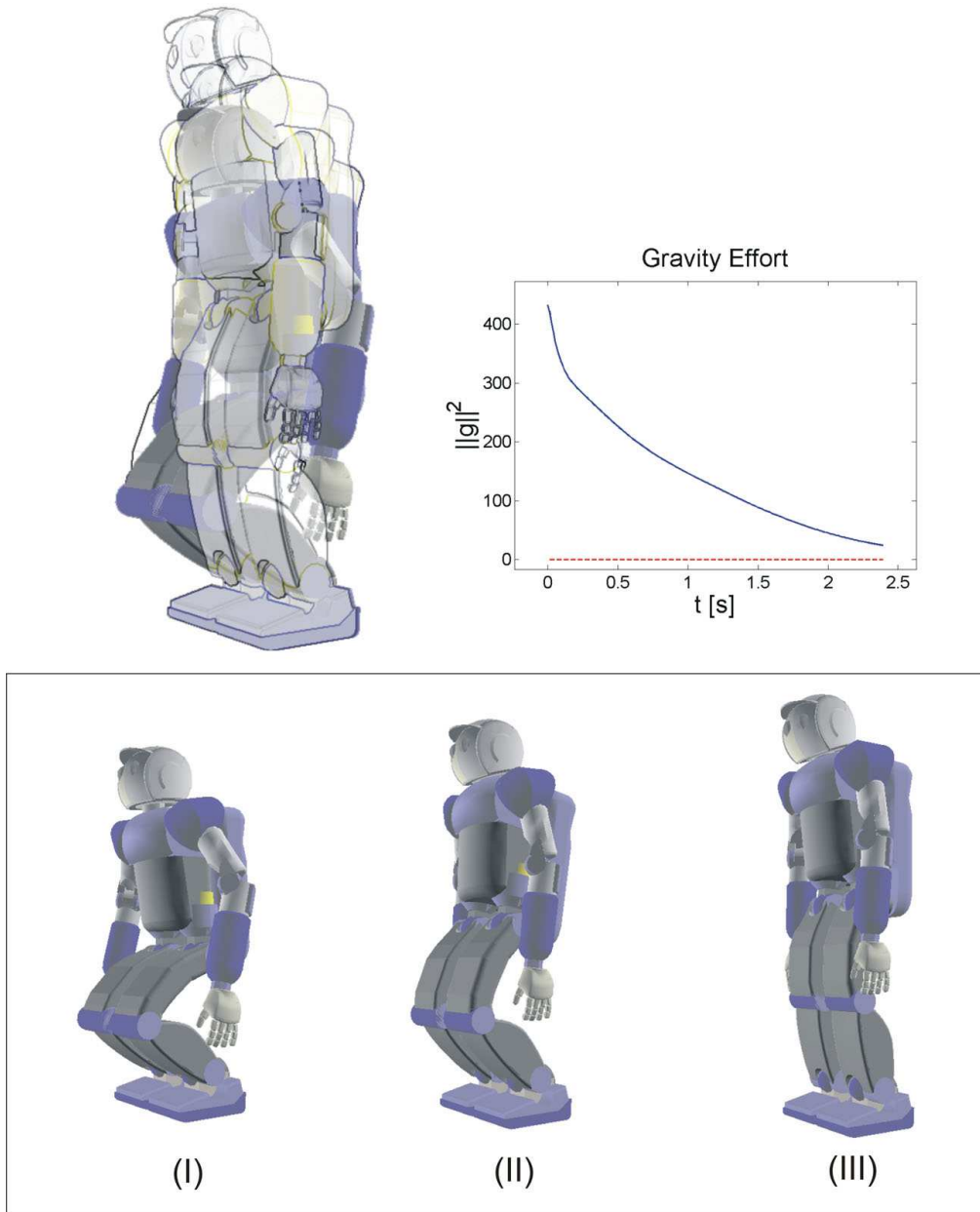


Figure 4.10: **Effort minimization while standing up:** This experiment validates the proposed whole-body effort minimization technique. The posture descends the effort potential until it reaches zero value.

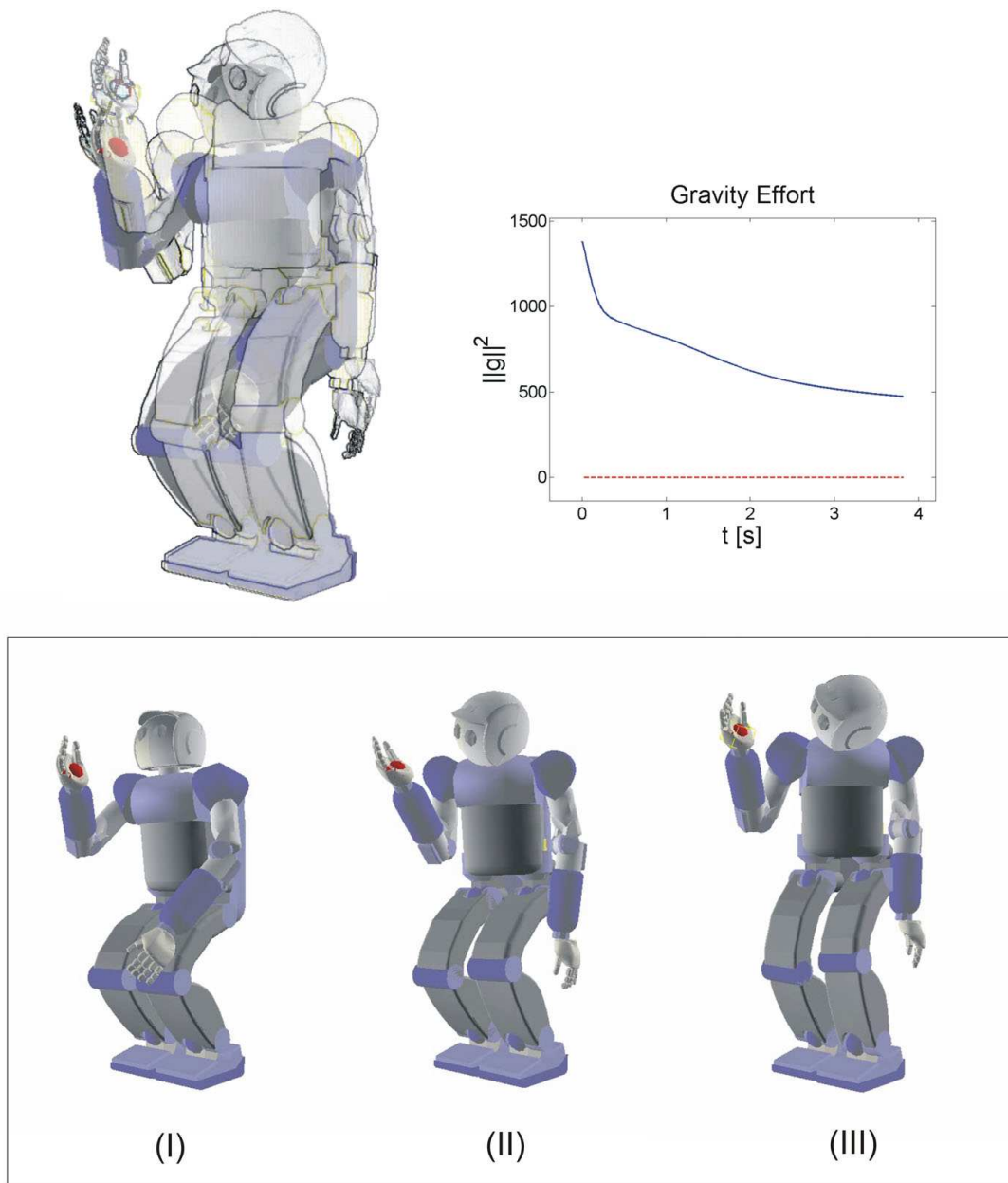


Figure 4.11: **Effort minimization under hand control:** Effort minimization is accomplished in the null-space balance and hand control. The robot's right hand is teleoperated to reach different points around the robot's workspace.

control is achieved using the same strategy than in the previous example.

At $t = 0$ the humanoid starts from a crouched pose while the right hand is teleoperated using the mouse. When posture control is activated, the robot's posture minimizes effort

energy. However, since the posture operates in the null-space of priority tasks, both COG and hand control are first accomplished. The residual redundancy is therefore used to minimize the effort. The data graph accompanying the figure show effort minimization while controlling the desired priority tasks. Once more the maximum posture velocity is $\omega_{\max} = 2\text{rad/s}$, the linear gain is $k_p = 1000$ and the velocity gain is $k_v = 2\sqrt{k_p}$.

4.4 Posture Stiffness Control

For tasks involving manipulation and locomotion behaviors, postural movement involves the heaviest parts of the robot's body, including the hips and the upper torso. In case of accidental collisions these parts can inflict great damage to the surrounding environment. Two aspects of the posture motion need to be controlled to provide maximum safety. First, joint velocities and accelerations of postural motion need to be limited below dangerous values. Second, the posture needs to be compliant upon external collisions without relying on force sensing.

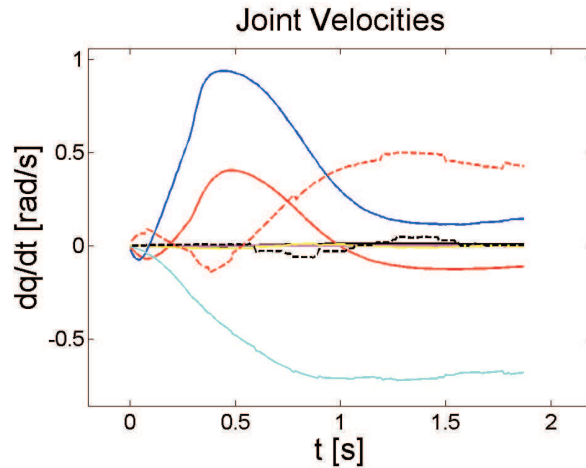


Figure 4.12: **Saturated joint velocities during effort minimization:** This data graph corresponds to joint velocities during the example shown in Figure 4.10.

Posture dynamics were characterized in (4.21) and a linearized control structure was achieved in (4.33). We will exploit these structures to implement safety procedures.

To limit maximum velocity and acceleration on the posture we propose using the following multi saturation control law first discussed in (3.47)

$$\alpha^{ref} = \nu_a \alpha_{des}, \quad (4.55)$$

$$\alpha_{des} = -k_v(\dot{q}_p - \nu_v \omega_{des}), \quad \nu_a = \min\left(1, \frac{\alpha_{max}}{\|\alpha_{des}\|}\right), \quad (4.56)$$

$$\omega_{des} = \frac{k_p}{k_v} \nabla V_p(q), \quad \nu_v = \min\left(1, \frac{\omega_{max}}{\|\omega_{des}\|}\right), \quad (4.57)$$

where α_{max} and ω_{max} are the maximum allowable acceleration and velocity values respectively, and ν_a and ν_v are saturation variables. When the acceleration component α_{des} reaches a value greater or equal to α_{max} the controller saturates accelerations. However, because posture control is linearized only within the controllable directions as shown in (4.33) the saturation behavior takes only place in these directions, i.e.

$$U_r^T \left(|\dot{q}_p| \leq \omega_{max} \frac{|\omega_{des}|}{\|\omega_{des}\|} \right), \quad (4.58)$$

$$U_r^T \left(|\ddot{q}_p| \leq \alpha_{max} \frac{|\alpha_{des}|}{\|\alpha_{des}\|} \right), \quad (4.59)$$

where $|\cdot|$ is the vector of absolute values of the enclosed term, and $\|\cdot\|$ is the Euclidian norm.

Let us consider once more the example on effort minimization shown in Figure 4.10. We set the following saturation values, $\omega_{max} = 1rad/s$ and $\alpha_{max} = 5rad/s^2$. The joint velocities for the right leg and upper body are shown in Figure 4.12. As we can see the velocities stay below the desired value and the accelerations are smooth. The highest values correspond to knee and hip joints.

Posture compliance is achieved by choosing low postural gains in the control law given in (4.55). In theory, the gains can be set as low as desired since we compensate for dynamic effects. In practice, we need to chose these gains to reject modeling errors. However, when comparing to inverse kinematic control methods our choice of gains is quite arbitrary whereas in inverse kinematic methods gains need to be very high. Our simulator for instance tolerates gains as low as $k_p = 20Nm/rad$ without loosing performance.

4.4.1 Examples: Response to Force Perturbations

We consider the example shown in Figure 4.13 involving mimicking an upright pose. The control gain k_p is set to a low value, $50Nm/rad$. We apply external forces on the robot's head and on his hips. The applied linear forces are of $300N$ in magnitude on the direction

pointed by the red arrows. As a result, the posture responds with high compliance, due to

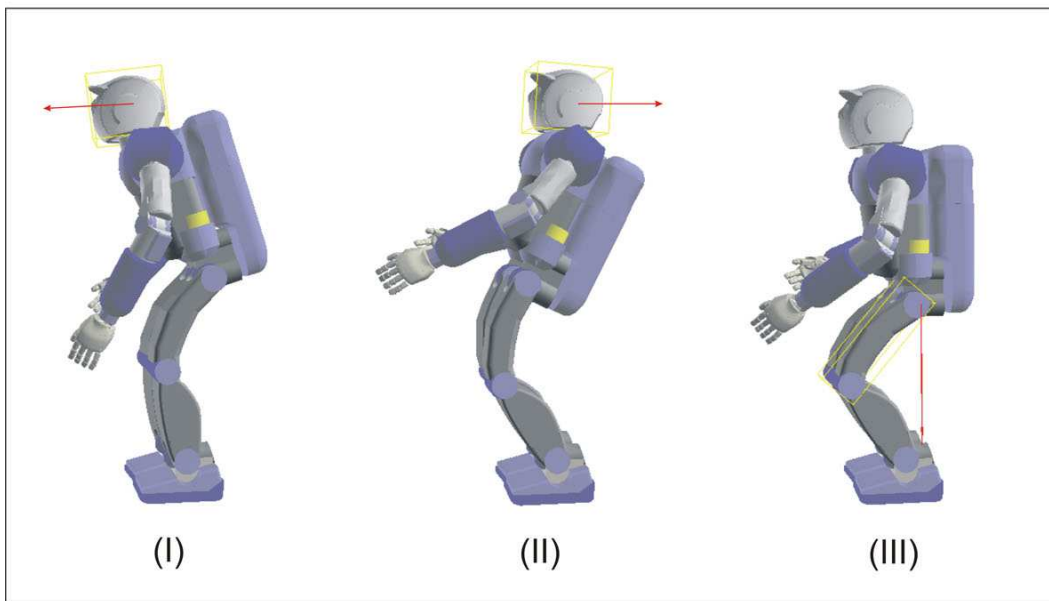
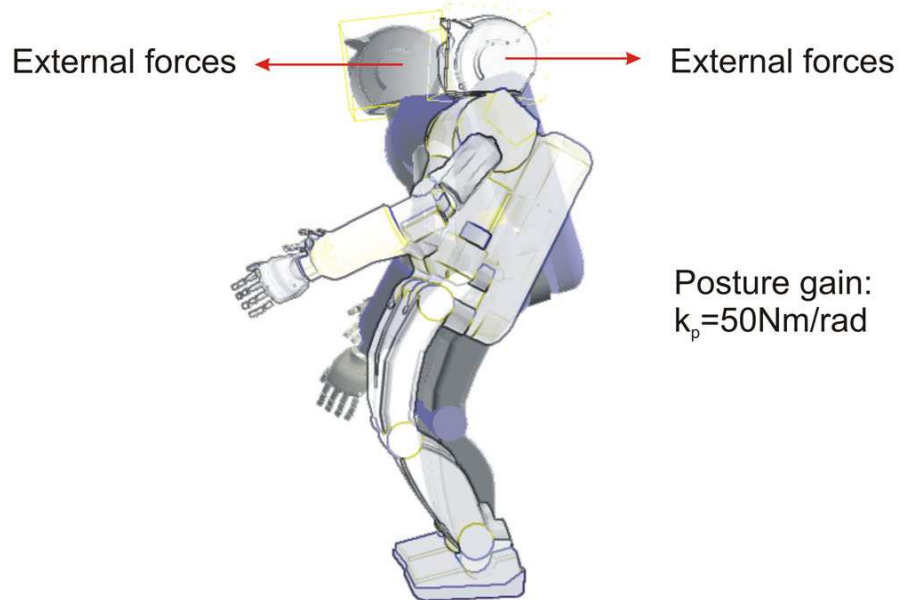


Figure 4.13: **Posture stiffness control:** This snapshots depict an actual experiment of posture compliance under low postural gains. The applied external forces are shown as red arrows.

the low stiffness gains. In fact, the whole-body behavior is compliant. When the applied forces are in the horizontal direction, the upper body complies on the x direction. When

the applied forces are in the vertical direction, the robot's legs comply downward on the z direction.

Chapter 5

Realtime Handling of Dynamic Constraints

We will present here reactive control methods to handle dynamic constraints in rapidly changing environments. When integrated into high level controllers or motion planners, these methods will allow robots to respond autonomously to dynamic events without interrupting the global task.

In control, a key problem concerns the combined interaction of high-level controllers with reactive methods. To execute sophisticated tasks, a global controller should be able to handle both local and global constraints. In this context, the goal of reactive controllers is to create immediate response to fast changing events while the goal of high-level controllers is to create a general behavioral response (emergent behaviors) or to find paths that achieve the desired task goals (motion planning). The goal of this chapter is to present reactive control methods to handle internal and external constraints during realtime interactions as well as to explore the connection of reactive techniques with high level controllers. We will focus on control methods to handle a variety of constraints including collision avoidance, joint limit constraints, self-collision avoidance, and support constraints.

Though realtime movement generation under dynamic constraints has been thoroughly studied in the context of mobile navigation, in humanoid systems this problem is starting to receive much attention. The objective of this chapter will be to develop techniques to handle dynamic events in humanoid systems. In the near future, coordinating reactive and high-level controllers will allow humanoids to engage in sophisticated tasks such as operating in tight spaces, walking among moving obstacles, and in general responding to a variety of dynamic events.

The methods we will describe here are aimed to support the implementation of high level controllers. For instance, our reactive techniques could support the creation of high level behaviors that would respond autonomously to internal and external constraints. Moreover, our control methods will provide the support to monitor task feasibility under all acting constraints allowing controllers to change robot behavior at runtime.

In the area of motion planning our control methods will provide new capabilities. Previous work in our lab concerns the development of the elastic strip framework (Brock and Khatib 2002), a dynamic planning method that can modify global paths in response to dynamic events. Although in its current state the elastic strip framework works well for mobile systems, it remains to extend it to humanoid systems where balance stability and supporting contacts are an integral part of the movement. The methods we will describe here will provide the necessary support to extend elastic strips to humanoids. For instance, our methods could be used to project candidate paths in the robot's posture space instead of relying on direct joint space mappings, providing support consistency and precise control of COG accelerations while tracking the desired paths. We will describe techniques that when connected to elastic planners will allow controllers to deform candidate paths not only in response to incoming objects but also in response to joint limits and self collisions.

With the methods we will describe here, whole-body control will be reduced to the planning and control of a few operational points regarding locomotion and manipulation tasks while balance stability, contact stance, and response to dynamic constraints will be automatically handled through reactive techniques. Our methods will be based on potential fields. For instance, collision avoidance will be implemented using repulsion fields while joint limits avoidance will be implemented using blocking attractors. Potential fields became popular in robotics after it was proposed by (Khatib 1986).

As part of this chapter, we will discuss the role of balance constraints. In previous chapters, manipulation and locomotion tasks were designed to operate in the null space of COG controllers. To handle all acting constraints, we will further project operational tasks in the combined null space of all acting constraints, thus preventing constraint violations. In Chapters 2 and 3 we presented control methods that provided linear control of COG accelerations. Linear control of COG accelerations will still be feasible in the presence of multiple acting constraints. Whole-body motion generation under balance constraints has been mostly studied as a planning problem (Kuffner et al. 2003; Hauser et al. 2006) or as a purely locomotion problem (Harada et al. 2004; Kajita et al. 2003b). Our approach will complement these methods by providing linear control of COG accelerations in the presence of multiple acting constraints.

Internal constraints such as self collision and joint limit avoidance are especially relevant when generating goal-based behaviors. Our approach to handle self collisions will rely on implementing repulsion fields on proximity points of nearby links, creating dual repulsion forces on link pairs. In our current implementation, to compute efficiently distances between link pairs we use a hierarchical sphere model (Ruspini and Khatib 1999) and apply Gilbert's efficient distance computation (Gilbert, Johnson, and Keerthi 1988). However, we are currently replacing these algorithms with a more efficient library called SWIFT++ (Ehmann and Lin 2000). Self collision constraints have been previously studied in the context of motion validation (Kanehiro and Hirukawa 2001; Kuffner et al. 2002). However, our control approach goes further ahead by providing support to modify the robot's pose in response to self collisions. When combined with local planners, our approach will provide the support to modify candidate paths in response to unplanned self-collision events.

To handle joint limits, our approach will consist on locking joints before they hit their limits. Strategies to handle joint limit constraints date back to (Liegeois 1977). With the implementation of visual servoing techniques, joint limit prevention has recently regained importance (Espiau, Chaumette, and Rives 1992; Marchand and Hager 1998). Our methods here will extend these approaches to operate in full humanoid systems, exploiting the overall system redundancy. In contrast with previous methods, our approach will rely on enforcing constraints as priority tasks while other operational tasks will operate in the residual redundancy. This technique will prevent operational tasks from violating constraints and will allow controllers to determine task feasibility under the acting constraints.

Collision constraints will be handled reactively via repulsion fields against incoming obstacles. Avoidance techniques have been popular in the context of path relaxation (Krogh 1984; Buckley 1986; Brock and Khatib 2002), high level reactive control (Khatib 1986; Brooks 1986), and collision free paths (Moravec 1980; Chatila 1981; Lozano-Perez 1983; Latombe 1991; Laumond and Jacobs 1994). Our techniques will enhance and complement previous reactive and non-reactive techniques.

This chapter is organized as follows. In Section 5.1 we will describe a novel control structure to handle constraints reactively. This structure will extend our previous work on multi-task control (see Chapter 3). In Section 5.2 we will characterized a variety of internal and external constraints and propose techniques to handle them in realtime. A large number of examples will be presented throughout the chapter.

5.1 Control Structure to Handle Constraints

In this section we will develop control techniques to respond to dynamic constraints while pursuing task goals. These techniques will be developed in the context of operational space control extending the whole-body control framework described in previous chapters.

Humanoids are aimed at executing realtime manipulation and locomotion tasks in complex environments, possibly with a high degree of autonomy. Operating in these environments entails responding to dynamic events such as moving obstacles and contact events without interrupting the global task. For instance, to get into a car a humanoid needs to handle a sequence of contact events while tracking a planned motion; balance stability and hip placement need to be simultaneously controlled. These type of control scenarios require control structures that can synthesize highly constrained movements. Our methods will provide this support.

We will develop new control structures that will operate with the whole-body controllers developed in previous chapters. Reactive response to movement constraints will be addressed as an analytical problem, without involving offline computations. In contrast with previous approaches, constraints will be handled as priority tasks, determining the feasibility of other tasks and shaping the robot's postural space.

5.1.1 Constraint Prioritization

Realtime response to motion constraints has been extensively addressed as a secondary process. In contrast, our approach will consist on handling motion constraints as priority processes and executing operational tasks in the null space of constrained tasks.

To illustrate our approach, let us consider the control example shown in Figure 5.1, where the robot's end-effector is commanded to move towards a target point. When no constraints are active, the end-effector is controlled using operational space control (Khatib 1987), i.e.

$$\Gamma = J_{\text{task}}^T F_{\text{task}}, \quad (5.1)$$

where Γ is the vector of actuation torques, F_{task} is a control force to move the end-effector towards the desired goal, and J_{task} is the end-effector's Jacobian matrix. When the elbow joint enters the activation zone (shown in red), we project the task in the constraint-consistent motion manifold, decoupling the task from the constraint. At the same time, an artificial attraction potential is implement to prevent the elbow from penetrating further into the activation area. The simultaneous control of constraints and operational tasks is

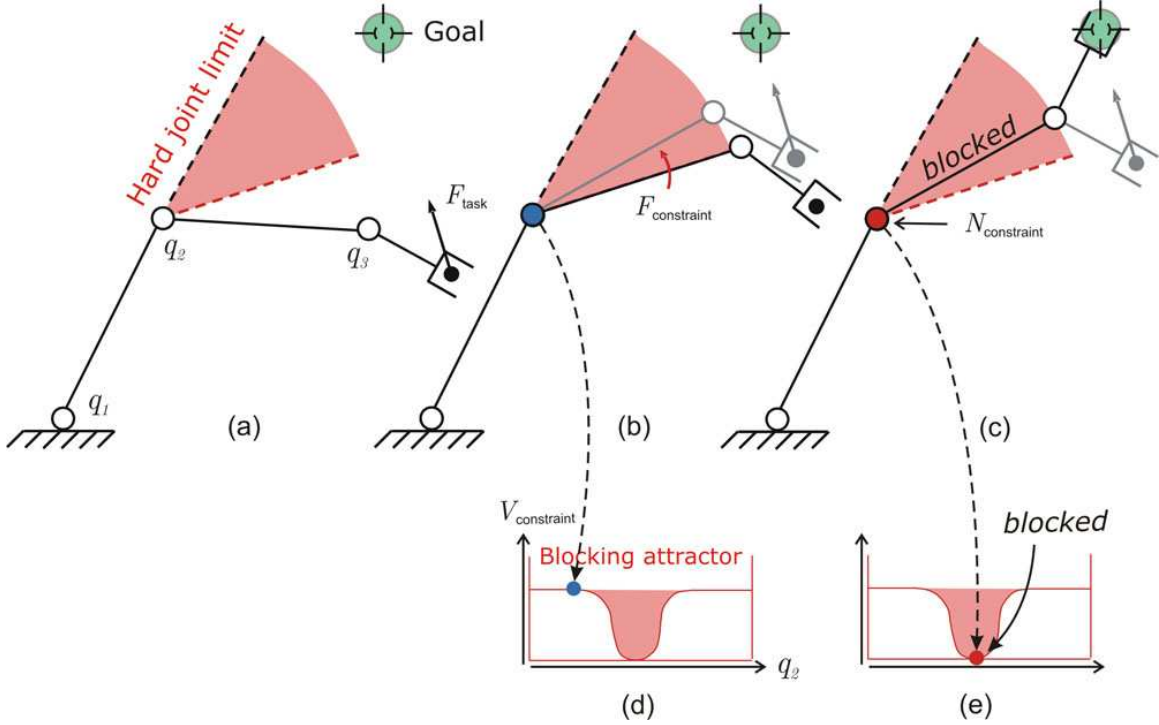


Figure 5.1: **Joint limits control concept:** In image (a), the robot’s end-effector has been commanded to move towards a desired goal. The red area defines a joint limit activation zone for the elbow joint. When this area is reached (b), a control approach is implemented to block the elbow joint while pursuing the goal (c). Images (d) and (e) depict the attractor potential used to block the elbow joint inside the activation area.

expressed as

$$\Gamma = J_{\text{constraint}}^T F_{\text{constraint}} + J_{\text{task|c}}^T F_{\text{task|c}}, \quad (5.2)$$

where

$$J_{\text{task|c}} \triangleq J_{\text{task}} N_{\text{constraint}} \quad (5.3)$$

determines a constraint-consistent projection of the end-effector task (the subscript $\{\text{task|c}\}$ indicates that the task is projected within the null-space of the constraint),

$$N_{\text{constraint}} \triangleq I - \bar{J}_{\text{constraint}} J_{\text{constraint}} \quad (5.4)$$

is the dynamically-consistent null space matrix of the constraint Jacobian, $F_{\text{constraint}}$ is the vector of blocking forces (in the example a 1D joint space torque) and will be soon characterized, $J_{\text{constraint}}$ is the Jacobian of the violating joint (in the example it would be a

constant matrix with zeros in non-violating joints and a one in the elbow joint), $F_{\text{task}|c}$ is the vector of task forces operating in the constraint-consistent motion manifold, and $\bar{J}_{\text{constraint}}$ is the dynamically consistent generalized inverse of the constraint Jacobian (Khatib 1987).

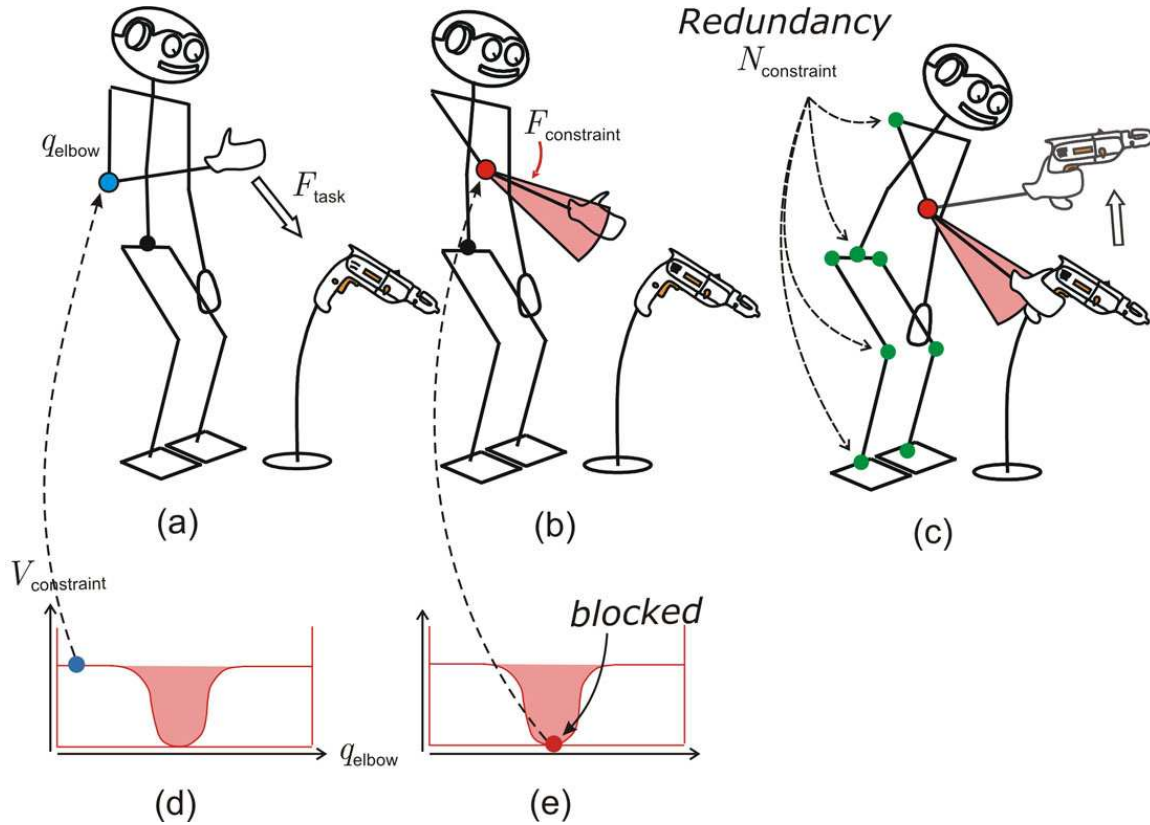


Figure 5.2: **Handling of joint limits on a humanoid:** This sequence of images depicts a robot reaching a target under joint limit constraints. In image (a), the robot's right hand is commanded to move towards the drill. In image (b), a constraint handling task is activated to prevent the right elbow to reach its hardware limit. In turn, the reaching task is projected into the null space of the constraint. In image (c), the robot reaches the drill while complying with the elbow constraint. When a new hand command is issued, the elbow joint is unlocked. A separate process is used to determine activation conditions.

When controlling full humanoid systems, the same prioritization principles than in (5.2) are applied. For instance, we consider the whole-body behavior illustrated in Figure 5.2, analogous to some manipulation behaviors presented in Chapter 3. The task decomposition to execute this behavior is shown below

Task Decomposition (Reaching a Drill)

Task Primitive	DOFs	Control Policy	Priority level
Balance	2 ($x - y$ plane)	position	1
Gaze orientation	2 (\perp plane)	position	2
Right hand control	6	hybrid	2
Whole-body posture	$n =$ number of joints	position	3

Using the multitask decomposition presented in (3.30), the control structure to accomplish the above behavior is

$$\Gamma = \left(J_{\text{balance}}^{*T} F_{\text{balance}} \right) + \left(J_{\text{tasks|p(2)}}^{*T} F_{\text{tasks|p(2)}} \right) + \left(J_{\text{postures|p(3)}}^{*T} F_{\text{postures|p(3)}} \right). \quad (5.5)$$

Here the subscripts $\{\text{task|p(priority)}\}$ indicate the task name and the priority order. In addition, the prioritized Jacobians of the above equation have the form given in (3.14) and the control forces have the form given in (3.26). Notice, that gaze and hand tasks operate with the same priority level. Therefore, they are combined into a single macro task as explained in (3.7) and (3.8). However, it would also be valid to control these two tasks with different priority levels.

As shown in Figure 5.2, when the right arm reaches full stretch, the right elbow joint enters the constraint activation zone. To prevent constraint violations, we project the entire whole-body control structure shown above into the constraint-consistent motion manifold while we lock violating joints according to the following control structure:

Proposition 5.1.1 (Constraint-consistent whole-body control). *The following control structure creates whole-body behaviors and provides a torque term to handle constraints while preventing constraint violations*

$$\Gamma = \left(J_{\text{constraint}}^{*T} F_{\text{constraint}} \right) + \left(J_{\text{balance|p(2)}}^{*T} F_{\text{balance|p(2)}} \right) + \left(J_{\text{tasks|p(3)}}^{*T} F_{\text{tasks|p(3)}} \right) + \left(J_{\text{postures|p(4)}}^{*T} F_{\text{postures|p(4)}} \right). \quad (5.6)$$

Here, the subscripts $\{\text{task|p(priority)}\}$ indicate the task name and the priority order. Notice also that we have committed to a predetermined hierarchy where balance tasks are controlled with lower priority than constraint-handling tasks, operational tasks are controlled with lower priority than balance tasks, and postural tasks are projected into the residual redundancy. The control forces $F_{\text{constraint}}$ will be used to handle the acting constraints.

Proof. The above control structure is an extension of the multi-task whole-body control structure shown in Theorem 4.1.1. Therefore it provides linear control of all prioritized tasks. The postural term above can be used to implement task based postures as shown in Proposition 4.2.1 or criteria based postures as shown in Theorem 4.3.1. \square

The above control structures will be used to handle a variety of constraints as will be discussed in Section 5.2.

The main goal of projecting operational tasks into the constraint consistent null space is not only to prevent constraint violations but also to provide the support to check behavior feasibility under the acting constraints. Checking task feasibility allows controllers to change behavior at runtime in response to dynamic constraints. Other methods to deal reactively with dynamic constraints project avoidance criteria in the task's null space, failing to prevent constraint violations during conflicting scenarios.

5.1.2 Realtime Response to Dynamic Constraints

We consider here potential field techniques to handle dynamic constraints in realtime. For example, let us analyze in more detail the joint limit behavior shown in Figure 5.2. When the elbow joint enters the constraint activation area implement the control structure shown in (5.6) and we apply blocking forces to stop the elbow joint inside the activation area. To lock the joint we use attraction fields as shown in Figure 5.3. This potential can be

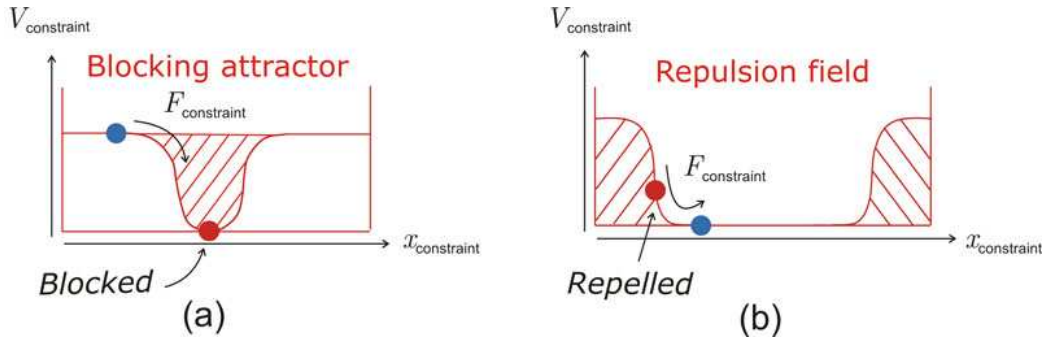


Figure 5.3: **Constraint handling potentials:** Image (a) depicts an attraction field that is used to lock joint limits when approaching hard limits, while image (b) depicts a repulsion field that is used to avoid obstacles or self-collisions.

expressed using the following energy function

$$V_{\text{constraint}} = \| q_{\text{elbow}}(t) - q_{\text{blocked}} \|^2, \quad (5.7)$$

where q_{elbow} represent the joint position of the elbow joint and q_{blocked} is the desired locked position. Furthermore, the constraint Jacobian consists of a simple selection matrix, i.e.

$$J_{\text{constraint}} = \begin{pmatrix} 0 \cdots 1 \cdots 0 \end{pmatrix} \quad (5.8)$$

where a 1 is placed to select the elbow joint, i.e.

$$\dot{q}_{\text{elbow}} = J_{\text{constraint}} \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}. \quad (5.9)$$

Here ϑ_b and \dot{q} are base and joint velocities as discussed in Chapter 2. In general, an arbitrary number of joint limit constraints can be handled by extending the previous potential function to multiple joints.

When handling obstacles and self collisions we use repulsion fields. A repulsion field is illustrated on image (b) of Figure 5.3. This potential can be expressed using the following energy function

$$V_{\text{constraint}} = \| d_{\text{obstacle}}(t) - d_{\text{safety}} \|^2, \quad (5.10)$$

where d_{obstacle} is the distance between the obstacle and the closest point on the robot's body and d_{safety} is a desired safety distance. The quantities are defined as follows

$$d_{\text{obstacle}} = x_{\text{robot}} - x_{\text{obstacle}}, \quad (5.11)$$

$$d_{\text{safety}} = K_{\text{safety}} \frac{d_{\text{obstacle}}}{\| d_{\text{obstacle}} \|}, \quad (5.12)$$

where x_{robot} is the Cartesian space position of the closest point to the obstacle on the robot's body, x_{obstacle} is the position of the closest point on the obstacle, and K_{safety} is a constant gain determining a safety margin. Though we define the above distances as 3D vectors, obstacle avoidance should be a 1D task acting on the direction of the distance vector. To map a 3D task into 1D space we manipulate the Jacobian associated with the distance vector, i.e. J_{robot} , removing unnecessary components, i.e.

$$J_{\text{constraint}} = \begin{pmatrix} {}^0R_d & S_n & {}^dR_0 \end{pmatrix} J_{\text{robot}}, \quad (5.13)$$

where dR_0 is a 3D rotation matrix between global frame and a frame aligned with the distance vector, and S_n is a 3×3 selection matrix that selects the components on the normal direction. Although the constraint Jacobian has three rows, its rank is one. As a result, when projecting constraint forces into actuation torques, only the perpendicular

direction to the obstacle will be considered.

To respond reactively to dynamic constraints we construct constraint handling tasks based on the above potentials. Handling constraints is therefore analogous to controlling operational tasks.

The dynamically-compensated structure to control the acting constraint is

$$F_{\text{constraint}} = \Lambda_{c|s} \ddot{a}_{\text{constraint}}^{\text{ref}} + \mu_{c|s} + p_{c|s}, \quad (5.14)$$

where $a_{\text{constraint}}^{\text{ref}}$ is the control policy to implement constraint potentials and will be described in a few lines. The above controller will yield the desired linear behavior

$$\ddot{x}_{\text{constraint}} = a_{\text{constraint}}^{\text{ref}}. \quad (5.15)$$

To implement the potentials shown in Figure 5.3 we use the velocity saturation control law previously shown in (3.43), i.e.

$$a_{\text{constraint}}^{\text{ref}} = -k_v (\dot{x}_{\text{constraint}} - \nu_v v_{\text{des}}), \quad (5.16)$$

$$v_{\text{des}} = \frac{k_p}{k_v} \nabla V_{\text{constraint}}, \quad \nu_v = \min\left(1, \frac{v_{\text{max}}}{\|v_{\text{des}}\|}\right), \quad (5.17)$$

5.1.3 Constraint-Consistent Task Control

Given the projection of whole-body behaviors shown in (5.6) how do we control operational tasks and postures given the acting constraints? Since constraints are handled via force level tasks as shown in (5.14), they can be directly integrated into our multi-task control framework described in (3.11) and (3.13).

Let us express the torque structure presented in (5.6) as

$$\Gamma = \Gamma_{\text{constraints}} + \Gamma_{\text{balance|p(2)}} + \Gamma_{\text{tasks|p(3)}} + \Gamma_{\text{postures|p(4)}}, \quad (5.18)$$

where each term $\Gamma_{\text{task|p(priority)}}$ represents the prioritized torque components shown in (3.11). Therefore, to control each prioritized task we use the prioritized operational space control structure presented in (3.20). For instance, based on the prioritized motion control structure shown in (3.26), balance under the acting constraints is controlled using the following

structure

$$\Gamma_{\text{balance|p(2)}} = J_{\text{b|p(2)}}^{*T} \left(\Lambda_{\text{b|p(2)}}^* a_{\text{balance}}^{ref} + \mu_{\text{b|p(2)}}^* + p_{\text{b|p(2)}}^* - \Lambda_{\text{b|p(2)}}^* J_{\text{b}} A^{-1} (S N_s)^T \Gamma_{\text{constraints}} \right), \quad (5.19)$$

where we have used the abbreviation $\{\text{b|p(2)}\} = \{\text{balance|p(2)}\}$. This structure will accomplish the desired linear behavior

$$\ddot{x}_{\text{balance}} = a_{\text{balance}}^{ref}. \quad (5.20)$$

To control balance (normally $x_{\text{balance}} = x_{\text{cogH}}$, i.e. the horizontal components of the robot's COG) we use the control coordinates shown in (2.88) and the control velocity and acceleration control laws shown in (3.47), (3.48), and (3.49). An important point here is that although the overall torque control structure can change when constraints kick in, the feedback control law a_{balance}^{ref} is independent of the constraints and therefore balance will not be disturbed.

Operational tasks besides balance control will be controlled using the control structure

$$\Gamma_{\text{tasks|p(3)}} = J_{\text{t|p(3)}}^{*T} \left(\Lambda_{\text{t|p(3)}}^* a_{\text{tasks}}^{ref} + \mu_{\text{t|p(3)}}^* + p_{\text{t|p(3)}}^* - (\Lambda_{\text{t|p(3)}}^* J_{\text{t}} A^{-1} (S N_s)^T (\Gamma_{\text{constraints}} + \Gamma_{\text{balance|p(2)}})) \right), \quad (5.21)$$

where we have used the abbreviation $\{\text{t|p(3)}\} = \{\text{tasks|p(3)}\}$. Once more, the above structure will yield linear control of operational tasks. For instance, for the example described in Figure 5.2 where the robot's head and right hand are controlled as an aggregated task as shown in Section 3.2.1, the above controller would result in the linear behavior

$$\ddot{x}_{\text{gaze}} = a_{\text{gaze}}^{ref}, \quad (5.22)$$

$$\ddot{x}_{\text{hand}} = a_{\text{hand}}^{ref}, \quad (5.23)$$

which has been accomplished using the aggregated acceleration vector, i.e.

$$a_{\text{tasks}}^{ref} = \begin{pmatrix} a_{\text{gaze}}^{ref} \\ a_{\text{hand}}^{ref} \end{pmatrix}. \quad (5.24)$$

Finally, postures will be controlled using all available motion redundancy using the structures presented in Proposition 4.2.1 and Theorem 4.1.1.

5.1.4 Task Feasibility Under Constraints

Constraint prioritization ensures that constraints are first accomplished and tasks operate in the constraint's redundant space. This projection provides the support to measure task feasibility and can be used to change overall behavior in response to the acting constraints.

To measure task feasibility, we propose one of the following quantities; the condition number of the prioritized Jacobian or the condition number of the inverse prioritized inertia matrix. Considering the structure presented in (5.18) and the task control structure presented in (5.21), task level condition numbers can be expressed as

$$\kappa\left(J_{\text{tasks|p}(3)}^*\right) = \frac{\sigma_1\left(J_{\text{tasks|p}(3)}^*\right)}{\sigma_r\left(J_{\text{tasks|p}(3)}^*\right)}, \quad \kappa\left(\Lambda_{\text{tasks|p}(3)}^{*-1}\right) = \frac{\sigma_1\left(\Lambda_{\text{tasks|p}(3)}^{*-1}\right)}{\sigma_r\left(\Lambda_{\text{tasks|p}(3)}^{*-1}\right)}, \quad (5.25)$$

where $\sigma_1(\cdot)$ and $\sigma_r(\cdot)$ are the first and last singular values of the enclosed term for an r -dimensional task. Both of these operators become singular when the task becomes infeasible under the acting constraints.

Furthermore, balance feasibility under the acting constraints should also be studied and monitored. This can be done by computing the condition numbers of the corresponding balance quantities, i.e.

$$\kappa\left(J_{\text{balance|p}(2)}^*\right) = \frac{\sigma_1\left(J_{\text{balance|p}(2)}^*\right)}{\sigma_2\left(J_{\text{balance|p}(2)}^*\right)}, \quad \kappa\left(\Lambda_{\text{balance|p}(2)}^{*-1}\right) = \frac{\sigma_1\left(\Lambda_{\text{balance|p}(2)}^{*-1}\right)}{\sigma_2\left(\Lambda_{\text{balance|p}(2)}^{*-1}\right)}. \quad (5.26)$$

Notice, that the balance task has two degrees of freedom, corresponding to the COG's horizontal coordinates.

An advantage of using the condition number of the inverse prioritized inertia over the condition number of the prioritized Jacobian is that the former has only dimension $r \times r$ where r is normally a small number, while the latter has dimension $r \times n$ where n is the number of actuated joints. However, to choose the best among the two we conducted the following experiment.

An example on task feasibility using our simulated humanoid robot Collabot is shown in Figure 5.4. This experiment has been conducted using the task decomposition proposed for the example of Figure 5.2. The robot's goal is to reach the red sphere without violating

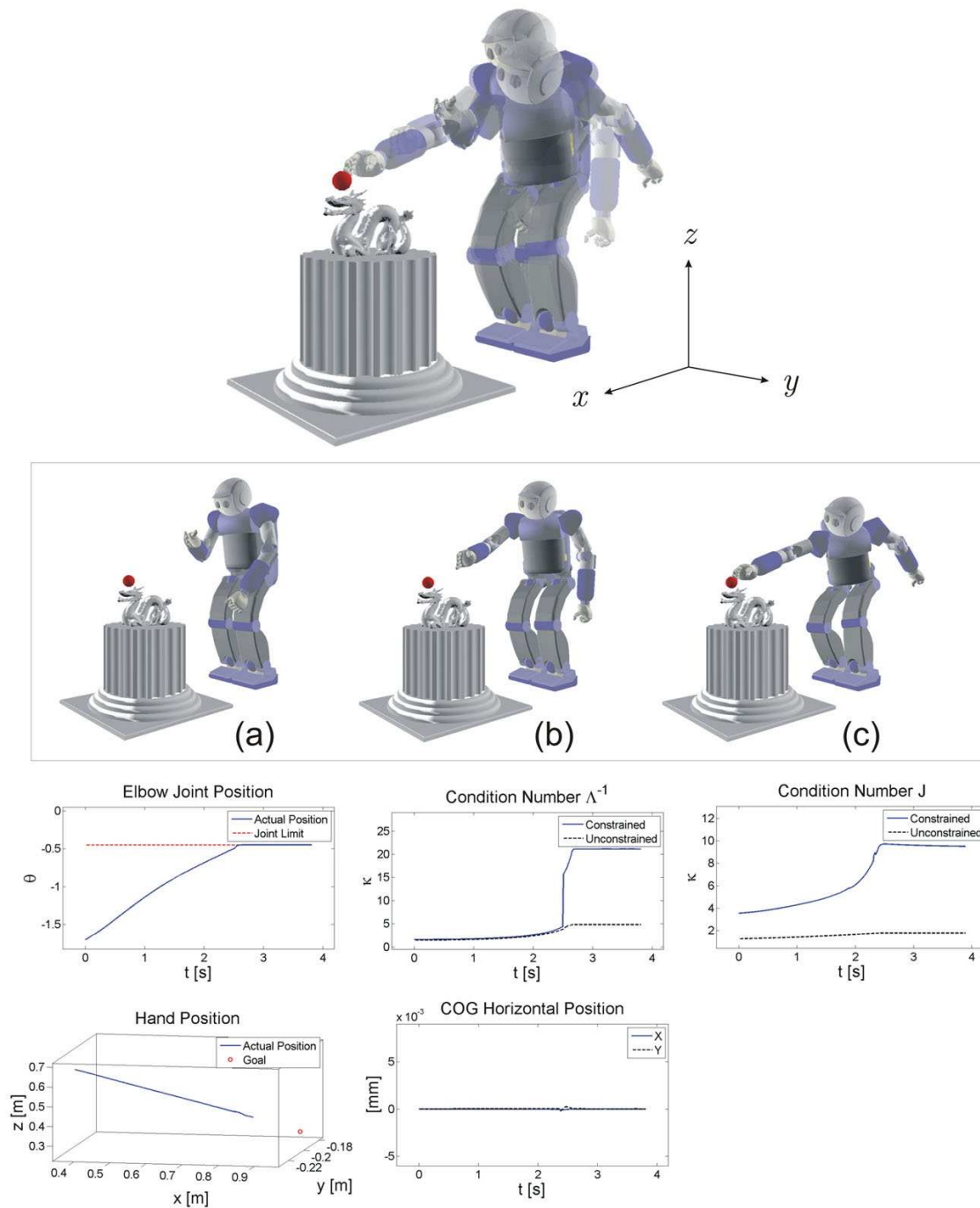


Figure 5.4: **Example on task feasibility:** Collabot is commanded to reach the red sphere, placed beyond the robot’s reachable workspace.

joint limits. Hardware joint limits for the elbow joint are reached when the upper and lower arm become aligned. The constraint activation zone is defined 0.5 rad before the hard limit. The controller described in (5.5) is used when no joint limits are active and the controller described in (5.6) is used when the constraint activation zone is reached. A velocity saturation control law with maximum velocity equal to 0.2 m/s is used to control the hand towards its goal.

The target point has been chosen in purpose to be out of reach and the behavior is not designed to initiate walking steps. Initially, the robot's right hand starts moving towards the goal and quickly reaches the maximum allowable velocity of 0.2 m/s . At $t = 2.5 \text{ s}$, the right arm has reached maximum stretch and as a result a constraint handling task is activated. Due to the action of the constraint, both the condition number of the constrained Jacobian $J_{\text{task}|p(3)}^*$ and the condition number of the prioritized inverse inertia matrix $\Lambda_{\text{task}|p(3)}^{*-1}$ grow rapidly towards infinity. We determine a cut off value for this last condition number equal to 20, which has been empirically chosen. When this value is reached, we change the command of the hand to stay at its current position. Notice that while the condition numbers of the prioritized (i.e. constrained) quantities grow rapidly towards infinity, the condition numbers of the corresponding unconstrained quantities stay within bounds. This characteristic validates the propose condition numbers as measurements of task feasibility. When the elbow joint reaches the constraint activation area at 0.5 rad a constraint handling task is activated to block the joint 0.05 rad inside. In the data graphs shown in Figure 5.4 we can observe that while the operational task is feasible its trajectory is completely straight (due to dynamic decoupling), and when it becomes infeasible due to balance and joint limit constraints it stops at its current position, several centimeters away from the target. Also, notice that the COG's horizontal position remains accurately centered with submillimeter error.

5.2 Types of Constraints and Control Approaches

In a near future, humanoids will be required to respond in realtime to a variety of dynamic constraints characteristic of human environments. In this section, we will discuss our approach to handle some important motion constraints including supporting contacts, balance, joint limits, moving obstacles, and self collisions.

5.2.1 Support Constraints

Supporting constraints were addressed in Chapter 2 assuming that the robot’s feet, and in general any other part of the robot’s body used for support are constrained by ground contacts. In Chapter 2 it was shown that to account for contact constraints we could formulate the following acceleration level equality constraint previously shown in (2.16)

$$\vartheta_s = 0 \quad (5.27)$$

$$\dot{\vartheta}_s = 0. \quad (5.28)$$

The impact of supporting constraint appeared first in the generalized equation of motion given in (2.24). When we developed multi-task control on Equation (3.20), contact constraints were integrated at the kinematic and dynamic levels. In particular, the Jacobian matrices of arbitrary tasks k were modified to integrate the contact constraints as well as the passive joints describing the global position and orientation of the robot’s in space as shown in (3.14).

These integration steps allowed us to project control structures directly in the space compliant with contact supports. As a result the proposed controllers would automatically assigned joint resources to accomplish the commanded accelerations while complying with the acting supports.

Let us study the example shown in Figure 5.5. The two sequences shown are part of an interactive posture behavior. Here Collabot is standing up with its right foot laying on top of a pedestal. The goal is to control the posture while maintaining contact constraints. To command interactively the robot’s posture we choose the following task decomposition,

Task Decomposition (Behavior with Foot on Pedestal)

<i>Task Primitive</i>	<i>DOFs</i>	<i>Control Policy</i>	Priority level
Balance	2 ($x - y$ plane)	position	1
Gaze orientation	2 (\perp plane)	position (goal-based)	2
Hip height	1 (z axis)	position (teleoperated)	3
Hip orientation	3	position (teleoperated)	4
Chest rotation	1 (around z axis)	position	5
Captured pose	n_{arms} = number of arm joints	position	6

Here the hip’s posture, which involves the control of the hip’s vertical height and orientation,

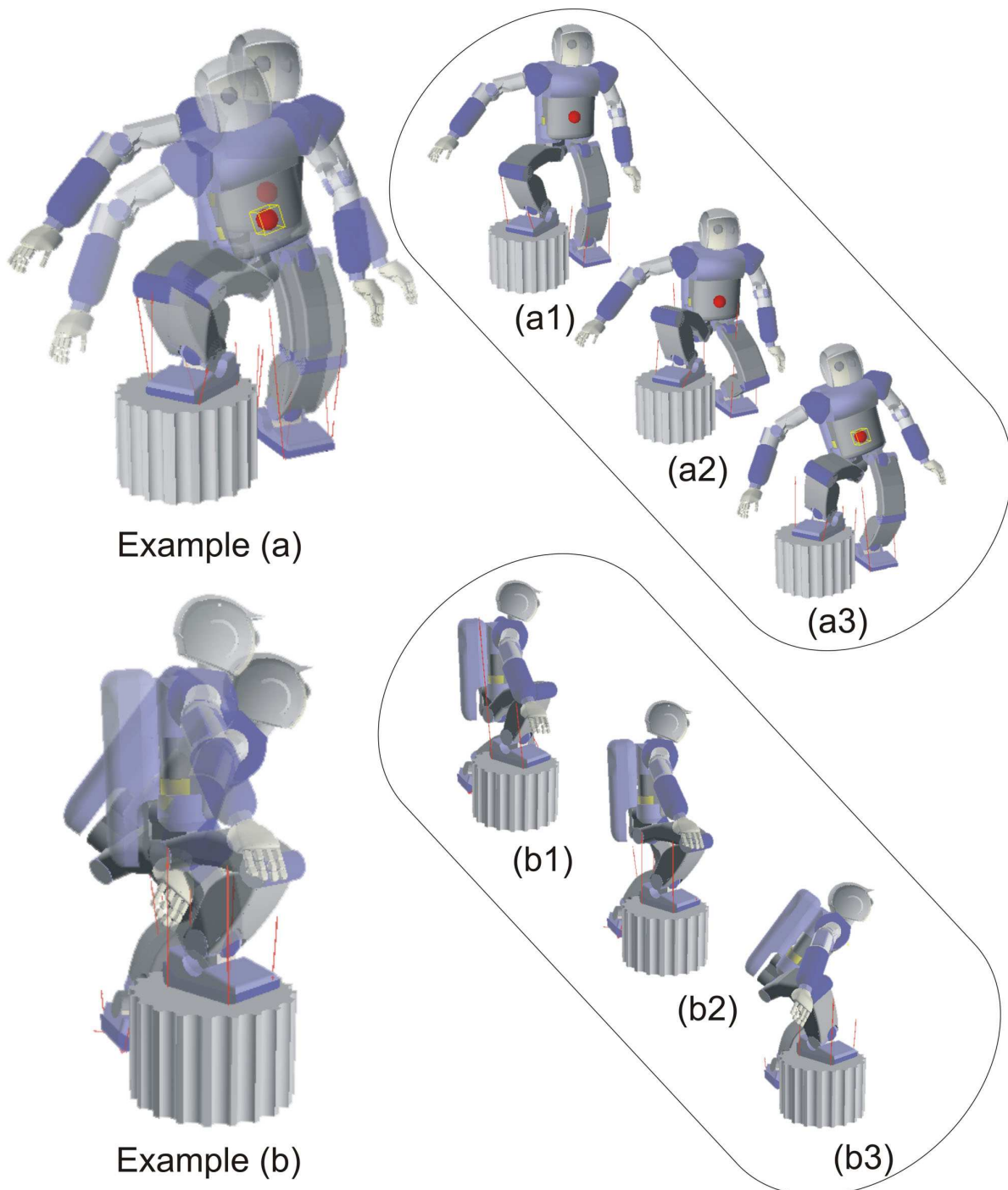


Figure 5.5: **Constrained behavior with foot on pedestal:** These two sequences of movement are part of an interactive posture behavior under arbitrary contact constraints. The red sphere is an interactive point that is modified at runtime by an operator. Support contacts are automatically handled as part of the control strategy

is teleoperated in realtime by manipulating the position and orientation of the red sphere shown in the figure. Vertical translations of the sphere will be translated into hip vertical positions while rotations along the sphere axes will be translated into hip rotations. The gaze is controlled to look at a fixed point in space. The torque actuation structure to simultaneously accomplish all goals is

$$\Gamma = \Gamma_{\text{balance}} + \Gamma_{\text{gaze|p}(2)} + \Gamma_{\text{hipHeight|p}(3)} + \Gamma_{\text{hipOri|p}(4)} + \Gamma_{\text{chestYaw|p}(5)} + \Gamma_{\text{armsPosture|p}(6)}, \quad (5.29)$$

where each priority task is controlled through the prioritized structure presented in (3.20).

As depicted in Figure 5.5, the desired operational and postural goals are accomplished independently of the robot's contact stance. This capability has been used throughout this dissertation to synthesize realtime behavior in a variety of contact scenarios.

5.2.2 Balance Constraints

Humanoids must keep balance using a small area defined by the supporting feet and other supporting contacts. The control of other operational tasks must be accomplished without compromising balance. As such balance acts both as a task and as a constraint. As a task, we are interested in determining whether balance can be maintained under the acting constraints. As a constraint, we are interested in determining whether manipulation and locomotion tasks can be accomplished without compromising balance stability.

To address these issues, our approach shown earlier in this chapter has been to create a prioritized control structure where balance acts with lower priority than internal and external constraints while other operational tasks and postures act with lower priority than balance, i.e.

$$\Gamma = \Gamma_{\text{constraints}} + \Gamma_{\text{balance|p}(2)} + \Gamma_{\text{tasks|p}(3)} + \Gamma_{\text{postures|p}(4)} \quad (5.30)$$

This ordering allows controllers to monitor balance and task feasibility under the acting constraints. Notice that this structure was already presented in (5.18). The control of balance can be done using the control structure presented in (3.65).

5.2.3 Obstacle Avoidance

To provide the support to operate humanoids in human environments, we need to develop control structures that can support the synthesis of avoidance behaviors. A great deal of work has been focused on the development of obstacle avoidance techniques in the context of both reactive and global control strategies (Maciejewski and Klein 1985; Khatib 1986; Latombe 1999; Brock et al. 2002; Kuffner et al. 2003).

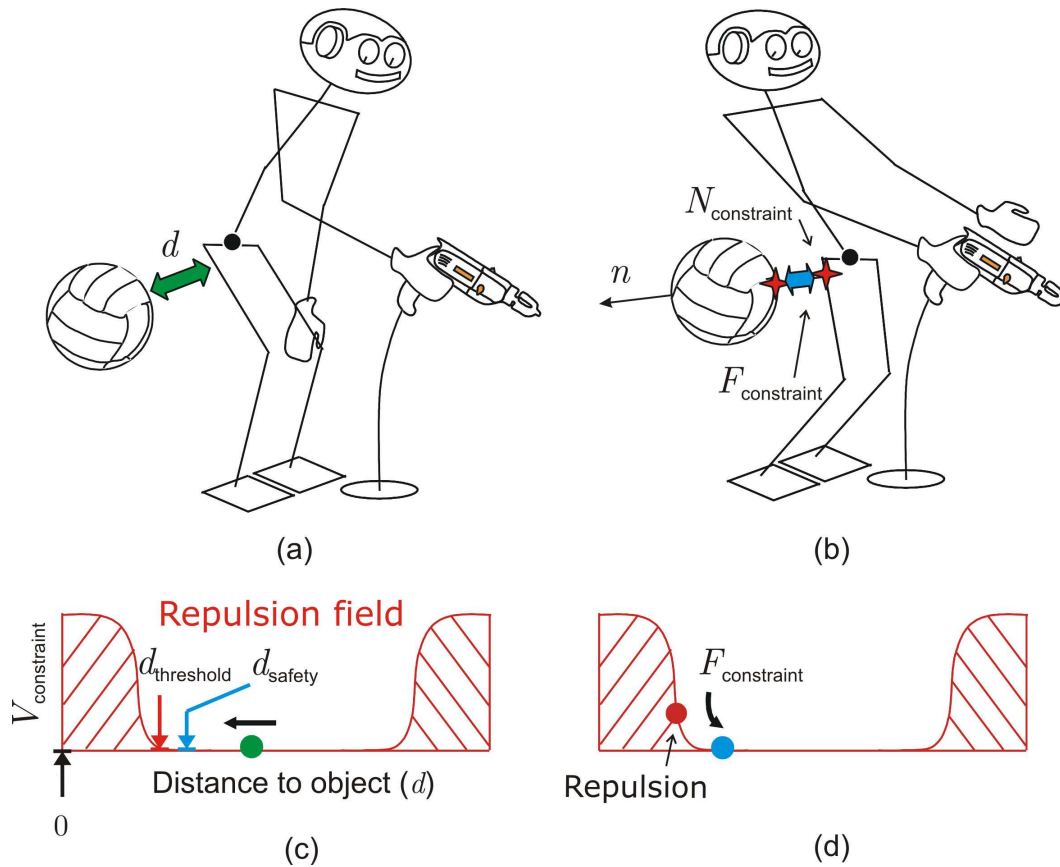


Figure 5.6: **Obstacle avoidance concept:** When an incoming obstacle approaches the robot's body a repulsion field is applied to the closest point on the robot's body. As a result, a safety distance can be enforced to avoid the obstacle.

The control structures presented at the beginning of this chapter are designed to respond quickly to incoming obstacles without interrupting the global task. Incoming obstacles can strongly shape the robot's free motion space. By implementing avoidance at the highest level, we can measure task feasibility given constraining obstacles. To handle obstacles we apply repulsion fields in the direction of the approaching objects as shown in Figure 5.6. Repulsion fields can be applied to desired points on the robot's body by using the control structure described in (5.14) and the velocity saturation control law described in (5.16).

Let us consider the example shown in Figure 5.7. The only task considered is balance stability, with no manipulation or locomotion tasks. The posture task is designed to imitate an upright pose. The objective is to demonstrate the response of a robot to incoming objects. A desired safety distance is set to 5cm . The same value is used as a threshold to activate an

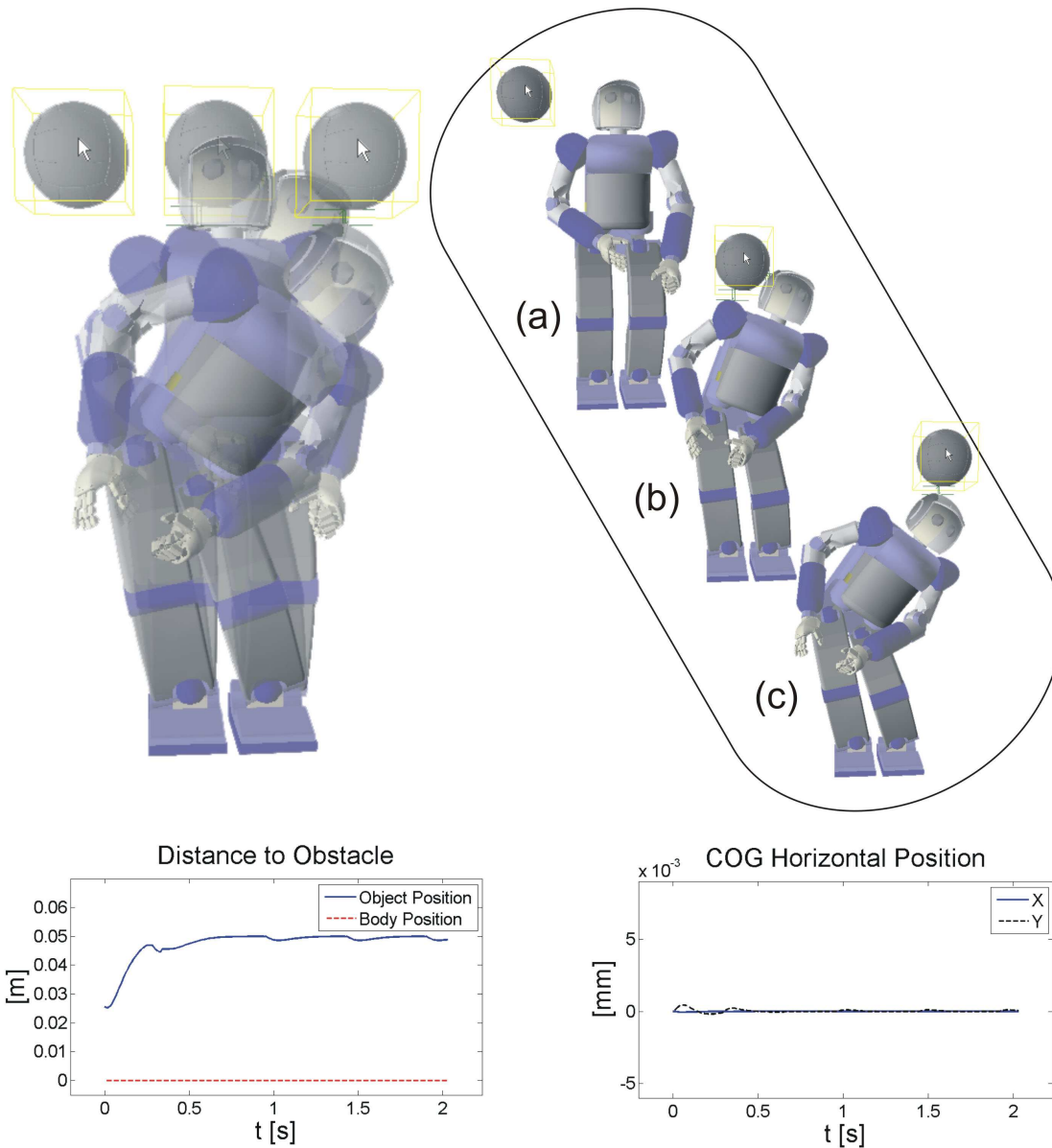


Figure 5.7: **Example on collision avoidance:** An obstacle approaches the robot’s body activating an avoidance task. The desired safety distance has been set to 5cm. The safety distance is approximately maintained while the error in balance remains very small.

avoidance task. When the obstacle crosses this threshold, the avoidance task is activated as a priority task while balance and posture control tasks are projected in the null space of the constraint. As we can observe, the desired safety distance is approximately maintained with respect to the moving obstacle without affecting COG control. Moreover, posture control is

also projected in the constraint's residual space allowing the robot to optimize poses while complying with the acting constraints.

A second example is shown in Figure 5.8. This time, not only balance stability and posture stance are actively controlled but also the robot's right hand position is controlled. Initially, an obstacle approaches the robot's head. Because there is enough available movement redundancy both the avoidance and the hand task can be simultaneously accomplished. However, when the obstacle approaches the robot's right hand, its position cannot be maintained. Using the feasibility indexes presented in (5.25) we can monitor this condition and remove hand control. The data graphs accompanying the figure depict the evolution of the condition number of the constrained task inertia. When the obstacle approaches the hand, this condition number grows rapidly towards infinity allowing the controller to remove the conflicting task.

5.2.4 Joint Limit Constraints

Because the handling of joint limits was analyzed in detail at the beginning of this chapter, in this section we will focus on two experiments involving goal-oriented control under multiple joint limit constraints.

At the beginning of this chapter we reviewed techniques to handle elbow joint constraints without interrupting the global task. As an extension, our approach to handle multiple joint limits is to construct a multi-joint task with individual locks for the violating joints. Let us define the joint position vector involving all violating joints, i.e.

$$q_c = \begin{pmatrix} q_i \\ q_j \\ q_k \\ \vdots \end{pmatrix}, \quad (5.31)$$

where i, j, k, \dots are violating joints. We define an attraction potential where each joint is attracted to a locking position inside their activation areas, i.e.

$$V_{\text{constraint}} = \| q_c - q_{\text{lock}(i,j,k,\dots)} \|^2, \quad (5.32)$$

where each violating joint has an associated lock position represented by the values $q_{\text{lock}(i,j,k,\dots)}$.

which corresponds to a selection matrix selecting components corresponding to violating joints. Using the control expression given in (5.14), the control law shown in (5.16), and the prioritized structure shown in (5.18) we can simultaneously handle multiple joint limits without interrupting the global task.

Let us study the example shown in Figure 5.9. The task consists once more on reaching a desired position shown as a red sphere, this time placed at ground level. Hip joint limits in Collabot are severe to prevent collisions between its bulky torso and its upper legs. Therefore the goal at ground level cannot be reached. At $t = 1$ s, the left hip roll joint is reached causing the controller to lock it. However, there is additional redundancy to proceed with the task. At $t = 2.5$ s the right hip pitch joint limit is reached and locked simultaneously with the left hip roll joint. Almost, simultaneously at $t = 2.6$ s the right elbow joint-limit is reached. With the right hip pitch and left hip roll joints locked, the knees cannot bend down. Therefore the task becomes unfeasible which is immediately reflected in the condition number of the hand's prioritized inverse inertia shown in the accompanying data graph.

A second example is shown in Figure 5.10 where the task corresponds to looking at a teleoperated point (shown as a red sphere). The two operational tasks here are to maintain balance and to control the robot's gaze. To maintain visual contact with the teleoperated point the robot uses all available joint resources. When joint limits are encountered they are locked and if there is additional movement redundancy the gaze task can still be controlled. The data graphs correspond to the sequence (a) and (b) of the figure where the robot is commanded to look downward. The head's pitch joint is first reached at $t = 1$ s. However, the task proceeds until the hip pitch joint limit is reached at $t = 2.7$ s. When the teleoperated point is further moved towards the robot's body there is no more available redundancy to continue looking. As a result the looking task becomes infeasible and the controller automatically halts the behavior.

5.2.5 Self Collision Avoidance

Self collisions are especially important in humanoid systems due to their high degree of mobility and anthropomorphic structure. Our approach to avoid self collisions is almost identical to avoiding obstacles. A potential field is created to maintain a safety distance between pairs of nearby links, i.e.

$$V_{\text{constraint}} = \| d_{\text{selfcollision}} - d_{\text{safety}} \|^2. \quad (5.34)$$

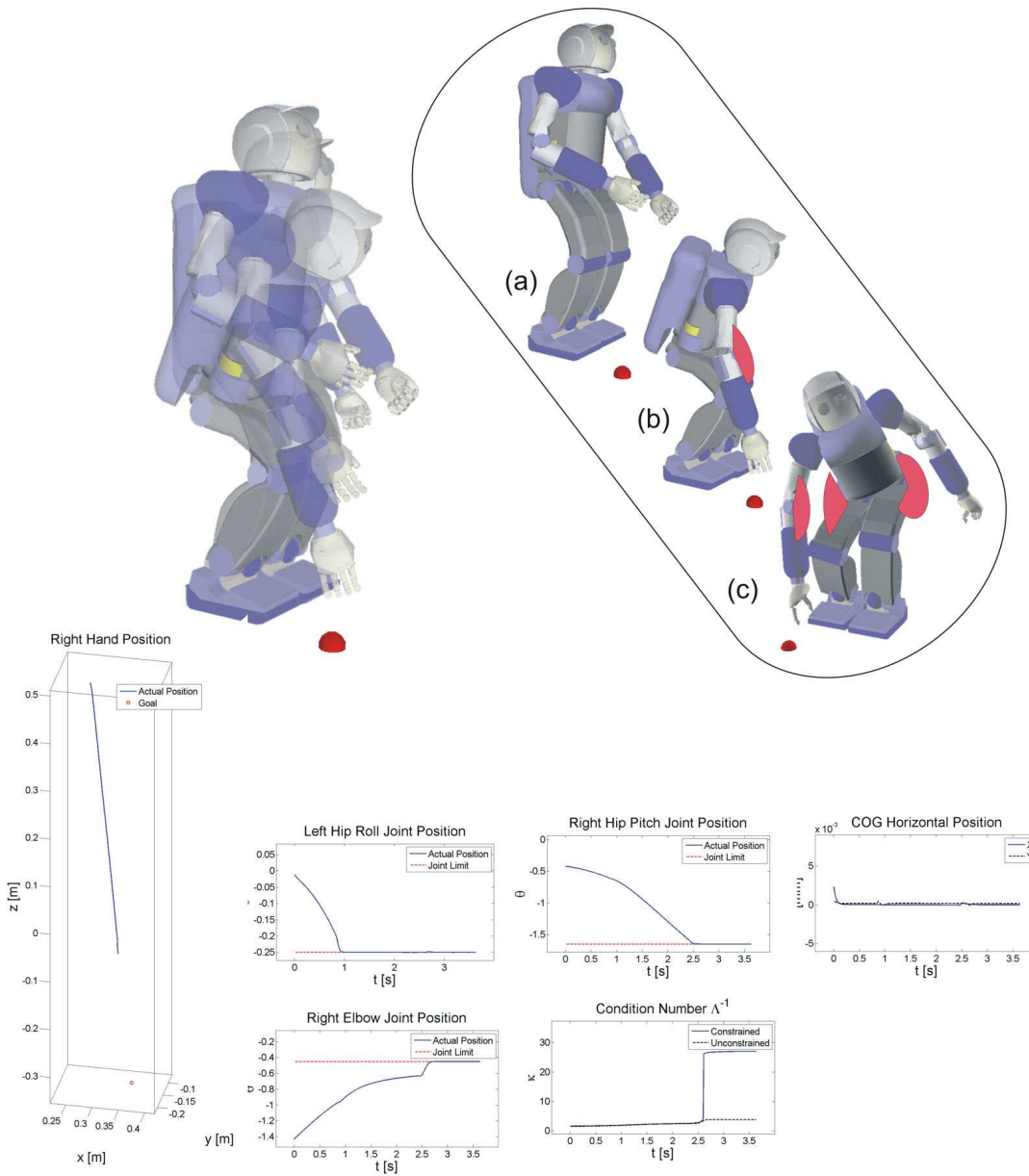


Figure 5.9: **Example involving multiple joint limit constraints:** In this example, Collabot is commanded to reach a desired point (sequence (a) and (b)). Image (c) corresponds to the same posture as shown in image (b) but from a different perspective. The red semicircles indicate the joints that are locked due to joint limit constraints. When the body achieves its maximum stretch the right elbow joint, the right hip pitch joint, and the left hip roll joint are locked to avoid reaching hard limits. As a result, the task becomes unfeasible and the right hand control is halted.

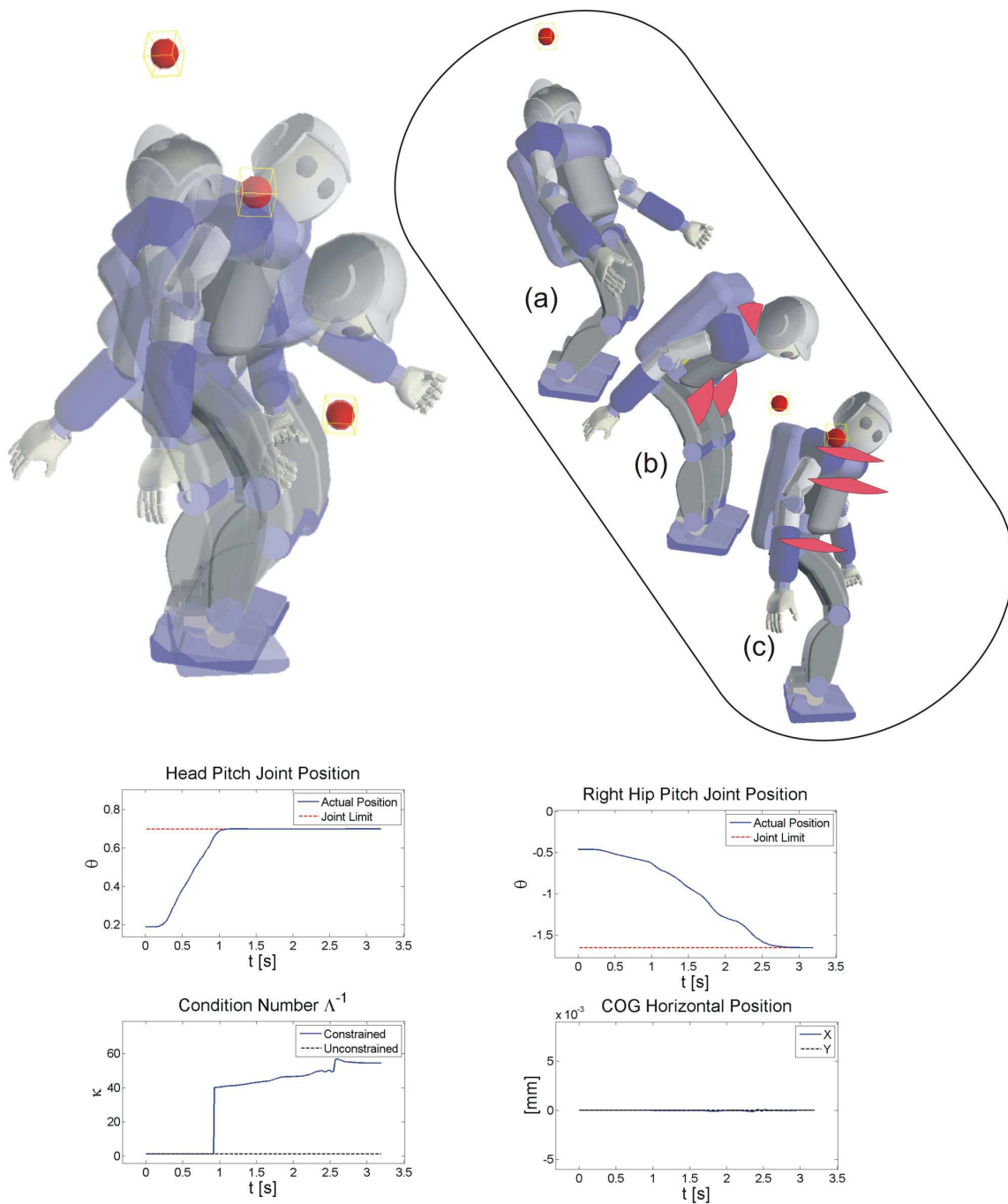


Figure 5.10: **Example involving head orientation under joint limits:** The red sphere corresponds to a teleoperated point that the robot is commanded to look at. All available joint resources are used to maintain sight of this point.

This time the distance vector corresponds to closest points on separate links, i.e.

$$d_{\text{selfcollision}} = x_{\text{link(a)}} - x_{\text{link(b)}}, \quad (5.35)$$

and the safety distance has identical form than for obstacle avoidance as shown in (5.11). Similarly to the Jacobian employed in collision avoidance shown in (5.13), we define the self collision Jacobian as

$$J_{\text{constraint}} = \left({}^0R_d \ S_n \ {}^dR_0 \right) J_{\text{distance}}, \quad (5.36)$$

where J_{distance} is the Jacobian associated with the previous distance vector. Once more, we rotate the 3D Jacobian of the distance vector to a frame that is aligned with the distance vector and we remove tangential components. Therefore, self collision avoidance is a 1D task, though its Jacobian has 3 rows. An example on self collision avoidance is shown in Figure 5.11. Here the right hand is teleoperated towards a position where the right arm intersects the robot's torso. A desired safety distance of 5cm is set between the robot's upper arm and the torso. When the arm crosses this threshold the proposed self collision avoidance task is activated. As a result the robot's torso rotates counterclockwise to avoid collision.

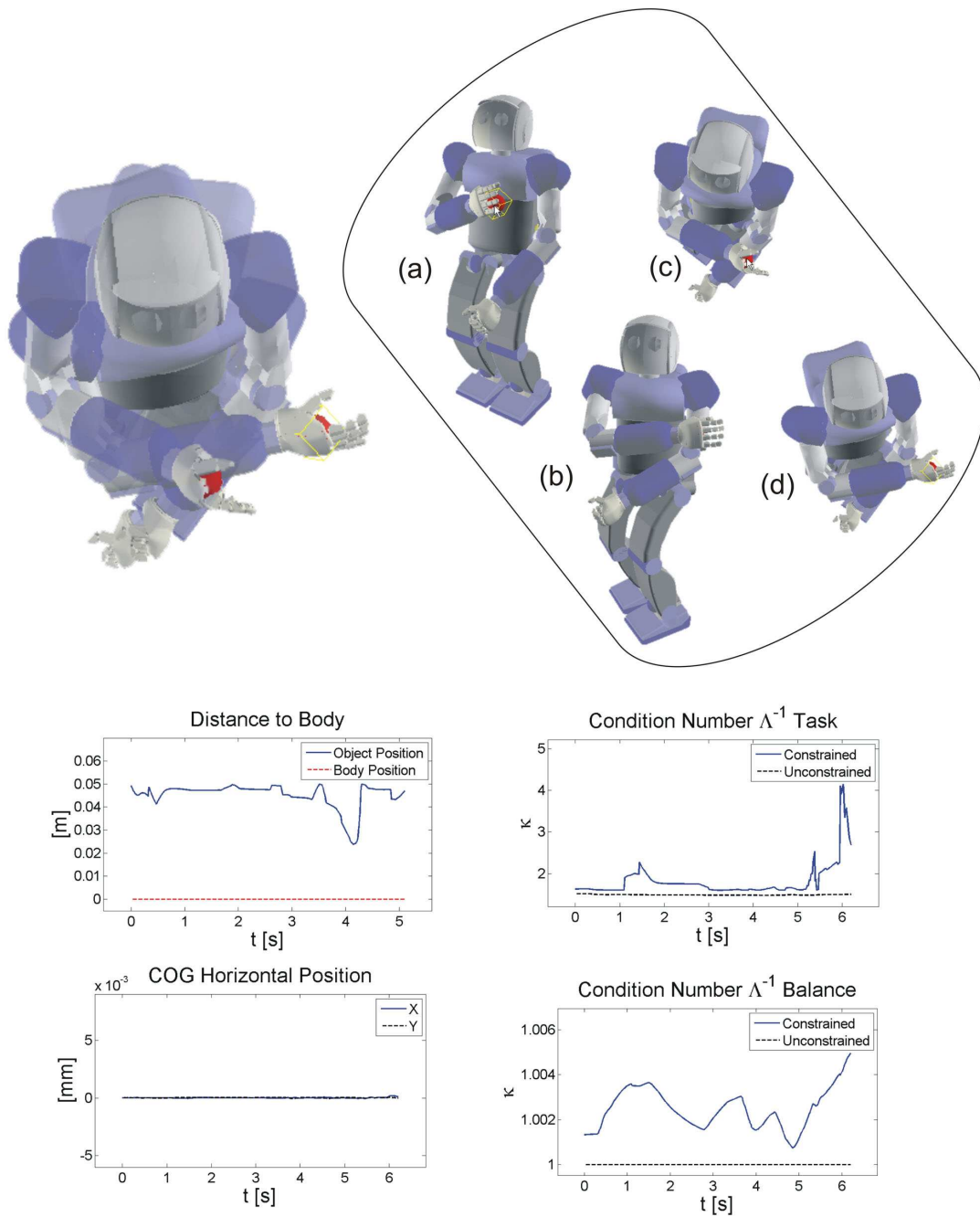


Figure 5.11: **Example on self collision avoidance:** The robot’s right hand is teleoperated towards a position where the right arm intersects the robot’s torso. The proposed self collision avoidance task is activated and as a result the robot’s torso rotates counterclockwise to avoid collision.

Chapter 6

Whole-Body Control of Movements in Midair

Besides realizing movements involving supporting limbs in contact with the ground, we expect humanoids to execute actions in midair during running and jumping behaviors or while performing acrobatic tricks. The goal of this chapter is to develop control methods to create movements in mid air. An example is shown in Figure 6.1 where a humanoid is commanded to jump upwards and stretch its legs into a straddle.

Behaviors involving mid air stages are increasingly being sought, and may become an important skill of humanoid systems. For instance, running is needed to provide fast locomotion means, intercepting objects in the air could be important for extreme interactions, and jumping is needed to overcome obstacles in unstructured terrain. More sophisticated acrobatic behaviors may involve all sorts of movements in the air, such as back flips, body twists, or somersaults.

When the robot loses contact with the ground there are no reaction forces acting on the robot's feet that can provide support to control the global center of gravity. If the robot's COG is considered as an inertial frame of reference, we can consider that its angular momentum is preserved in a weightless space while the COG draws a parabolic trajectory. These effects are characteristic of free flying and free floating systems.

The study of free flying and free floating systems has received much attention since the late 1980s with the advent of space robotics programs. Small spacecrafts with attached manipulators have been envisioned, developed, and tested (Oda, Kibe, and Yamagata 1996). In this context, several control frameworks have been proposed to control free-floating and free-flying systems in space. In (Umetami and Yoshida 1989) the notion of free floating

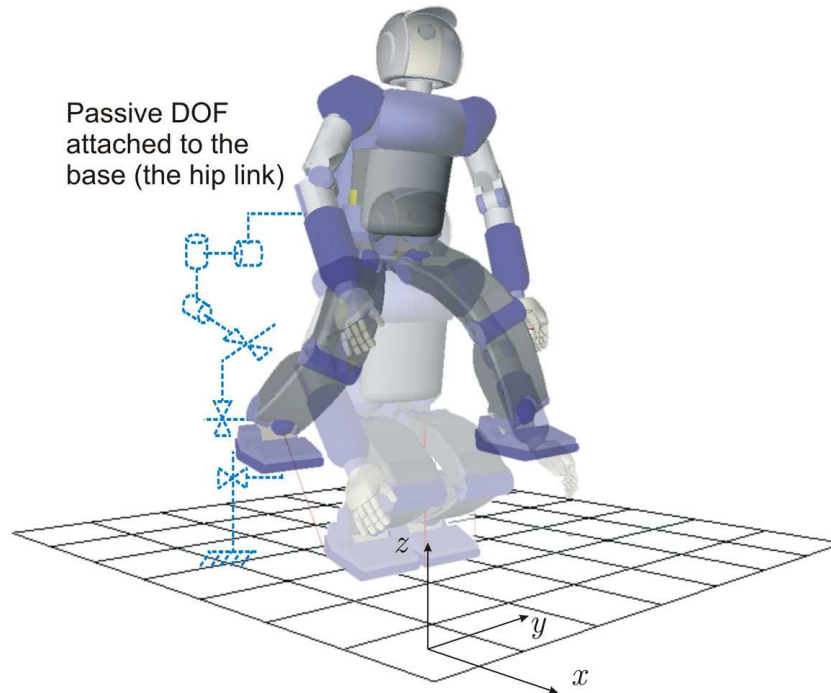


Figure 6.1: **Straddle jump:** This overlaid sequence taken from an actual simulated experiment shows a jumping sequence with a straddle motion. The COG's vertical trajectory is directly controlled in tasks space to lift the robot's body above the ground. Upon losing ground contact, a whole-body control strategy based on the methods we will discuss in this chapter is implemented. The feet are commanded to stretch out horizontally in a straddle and then to return to a landing position. Upon landing, the robot's balance can be regained.

Jacobian, a Jacobian matrix that describes the instantaneous kinematics of the robot under conservation of momenta, was introduced and later extended in (Nenchev, Umetami, and Yoshida 1992). In (Papadopoulos and Dubowsky 1991) the dynamics of free-floating systems were analyzed and torque controllers for space robots were proposed. In (Jain and Rodriguez 1993) fast algorithms for the computation of kinematic and dynamic quantities of under-actuated systems were proposed. In (Russakov, Rock, and Khatib 1995) the operational space formulation was extended to handle closed loop constraints in free-flying robots .

Other studies, have addressed the control of conventional manipulators with passive DOFs. In (Arai and Tachi 1991) control of robots with passive DOFs in operational space was studied, and later extended to contact tasks (Arai and Khatib 1994).

More relevant to our work, the study of legged robots that could run, trot, hop, even execute somersaults was pioneered by (Raibert 1986) in the MIT Leg Lab. More recently,

simulation and control of robots that can trot has been studied by (Palmer and Orin 2006).

In this chapter we will extend several of the reviewed techniques and develop a whole-body control framework to simultaneously control multiple operational tasks as well as the robot's posture during movements in mid air.

During movements in midair, the robot's center of gravity cannot be controlled, therefore our whole-body control methods will be used to anticipate landing, intercept objects in the air, and perform acrobatic tricks.

This chapter is organized as follows. In Section 6.1 we will review the basic equations of motion and physics of robots in free space. In Section 6.2, we will characterize task kinematic and dynamic representations in free space and extend the operational space formulation (Khatib 1987) to free floating systems. In Section 6.3 we will develop prioritized controllers to control multiple tasks and postures in free space. Finally, in Section 6.4 we will discuss several examples.

6.1 Basic Equations of Motion and Physics of Free Space

While in mid air, conservation of angular momentum imposes dynamic constraints on the robot's motion that need to be characterized to develop whole-body controllers.

6.1.1 Joint Space Dynamics

We consider the jumping example shown in Figure 6.1. The base position and orientation and the associated velocities are characterized by vectors

$$x_b = \begin{pmatrix} x_{b,p} \\ x_{b,r} \end{pmatrix}, \quad \vartheta_b = \begin{pmatrix} v_b \\ \omega_b \end{pmatrix}, \quad (6.1)$$

as shown in (2.1) and (2.2). The dynamic behavior of the robot in free space is characterized by the following equation of motion equivalent to (2.4) but with no reaction forces

$$A \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} + b + g = S^T \Gamma. \quad (6.2)$$

where q is the vector of joint coordinates, S is the actuation matrix shown in (2.19), and Γ is the $n \times 1$ vector of actuation torques. Moreover, A , b , and g correspond to the inertia matrix, Coriolis/centrifugal forces, and gravity forces of the free floating system respectively. It is also convenient to characterize the dynamic behavior of actuated joints with respect to

actuation torques which can be obtained by left-multiplying the above equation of motion by the transpose of the dynamically weighted generalized inverse of S , defined further below.

Definition 6.1.1 (Free floating inertia matrix). *The following matrix determines the inertial behavior of actuated joints*

$$A^* \triangleq \left(SA^{-1}S^T \right)^{-1}. \quad (6.3)$$

This matrix was first characterized in (Xu and Kanade 1992) and (Yoshida 1994) under the name of generalized inertia matrix.

Lemma 6.1.1 (Dynamically consistent generalized inverse of S). *A generalized inverse of S that projects the equation of motion shown in (6.2) into actuated space is the dynamically weighted generalized inverse of S with weight equal to A^{-1} , i.e.*

$$\bar{S} \triangleq A^{-1}S^TA^*. \quad (6.4)$$

Proof. Left multiplying (6.2) by the transpose of the above expression leads to the following reduced equation of motion

$$A^*\ddot{q} + b^* + g^* = \Gamma, \quad (6.5)$$

where A^* is the free floating inertia shown in (6.3), and

$$b^* = \bar{S}^T b, \quad (6.6)$$

$$g^* = \bar{S}^T g \quad (6.7)$$

are Coriolis/centrifugal and gravity terms respectively. Here we have used the equalities

$$\dot{q} = S \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad \ddot{q} = S \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix}. \quad (6.8)$$

Because this equation of motion characterizes the dynamic behavior of actuated joints with respect to actuation torques, \bar{S} corresponds to the dynamically consistent generalized inverse of S . \square

Property 6.1.1 (Commutation of $\bar{S}S$ with respect to A^{-1}). *The following equalities hold,*

$$\bar{S}SA^{-1}S^T\bar{S}^T = A^{-1}S^T\bar{S}^T = \bar{S}SA^{-1}. \quad (6.9)$$

Proof. Using the expression of \bar{S} given in (6.4) and the expression of A^* given in (6.3) we can write the following equations

$$\bar{S}SA^{-1}S^T\bar{S}^T = A^{-1}S^T A^*(A^*)^{-1}A^*SA^{-1} = A^{-1}S^T A^*SA^{-1} = A^{-1}S^T\bar{S}^T. \quad (6.10)$$

The reciprocal can be demonstrated following similar steps. \square

Let us also consider the following block decomposition of dynamic quantities,

$$A = \begin{pmatrix} A_{bb} & A_{br} \\ A_{rb} & A_{rr} \end{pmatrix}, \quad b = \begin{pmatrix} b_b \\ b_r \end{pmatrix}, \quad g = \begin{pmatrix} g_b \\ g_r \end{pmatrix}, \quad (6.11)$$

where the subscripts containing b and r indicate passive and active (i.e. base and robot) components, and the mixed subscripts correspond to coupling components.

Property 6.1.2 (Alternative expressions of free floating quantities). *The following expressions are equivalent to the dynamic quantities shown in Equations (6.3), (6.6), and (6.7)*

$$A^* = A_{rr} - A_{rb}A_{bb}^{-1}A_{br} \quad (6.12)$$

$$b^* = b_r - A_{rb}A_{bb}^{-1}b_b \quad (6.13)$$

$$g^* = g_r - A_{rb}A_{bb}^{-1}g_b. \quad (6.14)$$

Proof. The expression of A^* will be demonstrated in (6.32). Using the expression of \bar{S} given in (6.1.2) it is straightforward to demonstrate the expressions of b^* and g^* . \square

6.1.2 Analysis of the System's Momentum

Characterizing the robot's momentum in free space will reveal dependencies between base and joint displacements. These kinematic dependencies will later be used to develop operational space controllers for movements in mid air. To study the system's momentum let us consider its Euler-Lagrangian representation, i.e.

$$\frac{d}{dt} \begin{pmatrix} \partial_b K \\ \dot{q} \end{pmatrix} - \frac{\partial(K - U)}{\begin{pmatrix} \partial \tilde{x}_b \\ \partial q \end{pmatrix}} = S^T \Gamma, \quad (6.15)$$

where K and U correspond to kinetic and potential energy values respectively. To match dimensions, the vector of base positions and orientations \tilde{x}_b is assumed to be expressed using Cartesian space positions and Euler angles. The kinetic energy of the system can be expressed in terms of the system's inertia as follows

$$K = \frac{1}{2} \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}^T A \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}. \quad (6.16)$$

This allows us to split (6.15) into passive and actuated terms, i.e.

$$\frac{d}{dt} \frac{\partial K}{\partial \vartheta_b} - \frac{\partial(K - U)}{\partial \tilde{x}_b} = 0, \quad (6.17)$$

$$\frac{d}{dt} \frac{\partial K}{\partial \dot{q}} - \frac{\partial(K - U)}{\partial q} = \Gamma. \quad (6.18)$$

Since the inertia matrix A is independent of the robot's position and orientation in space, the partial derivative of the kinetic energy with respect to \tilde{x}_b is equal to zero, i.e. $\partial K / \partial \tilde{x}_b = 0$. On the other hand, the robot's potential energy U is due to gravitational forces. The partial derivative of U with respect to \tilde{x}_b corresponds to the first six components (the base components) of the gravity term g shown in (6.11), i.e.

$$g_b = -\frac{\partial U}{\partial \tilde{x}_b}. \quad (6.19)$$

Developing the robot's kinetic energy given in (6.16) we obtain the equality $K = \vartheta_b^T A_{bb} \vartheta_b + \dot{q}^T A_{rb} \vartheta_b + \vartheta_b^T A_{br} \dot{q} + \dot{q}^T A_{rr} \dot{q}$, which allows us to further transform (6.17) into the equality

$$\frac{d(A_{bb} \vartheta_b + A_{br} \dot{q})}{dt} + g_b = 0. \quad (6.20)$$

Integrating this equation reveals the system's momentum, i.e.

$$A_{bb} \vartheta_b + A_{br} \dot{q} + \int_0^t g_b d\tau = L_{\text{initial}}. \quad (6.21)$$

where L_{initial} is the initial angular momentum. The above expression allows us to write the following dependency between base and joint velocities

$$\vartheta_b = -A_{bb}^{-1} A_{br} \dot{q} + A_{bb}^{-1} L_{\text{initial}} - A_{bb}^{-1} \int_0^t g_b d\tau. \quad (6.22)$$

6.1.3 Constrained Kinematics

The previous dependency between base and joint velocities will allow us to represent kinematic behavior of arbitrary points on the robot's body as a function of joint velocities. These representations will be used to develop operational space controllers as well as to develop strategies to measure task feasibility under actuator commands. For an arbitrary point x on the robot's body, we consider the following velocity representation

$$\dot{x} = J \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} \quad (6.23)$$

where J is the full task Jacobian. The following expression of J reveals the contribution from the passive chain (the virtual joints describing the movement of the base) and the actuated joints

$$J = \begin{pmatrix} V_b & J_r \end{pmatrix}, \quad V_b \triangleq \begin{pmatrix} I & \hat{p}_{b,x} \\ 0 & I \end{pmatrix} \in \mathcal{R}^6, \quad (6.24)$$

where V_b is a transformation matrix which maps angular movement of the base to linear velocities at the task point (see discussion on macro/mini structures in Khatib 2004), J_r corresponds to displacements of actuated joints of the robot with respect to its base, $p_{b,x}$ corresponds to the distance vector between the task point and the robot's base, $\hat{p}_{b,x}$ is the cross product operator associated with the position vector.

By replacing ϑ_b in (6.23) by the constrained term of (6.22), we reveal the dependency of task velocities with joint velocities, i.e.

$$\dot{x} = J \begin{pmatrix} -A_{bb}^{-1} A_{br} \\ I \end{pmatrix} \dot{q} + J \begin{pmatrix} A_{bb}^{-1} L_{\text{initial}} - A_{bb}^{-1} \int_0^t g_b d\tau \\ 0 \end{pmatrix}. \quad (6.25)$$

Using the decomposition given in (6.24) we can further express the above velocity as

$$\dot{x} = J^* \dot{q} + \dot{x}_{\text{bias}}, \quad (6.26)$$

where

$$J^* \triangleq J_r - V_b A_{bb}^{-1} A_{br} \quad (6.27)$$

is referred to as the free floating Jacobian (Umetami and Yoshida 1989; Papadopoulos and Dubowsky 1991) and will be further characterized below, and x_{bias} corresponds to the rightmost term of (6.25).

Lemma 6.1.2 (Free floating Jacobian). *The following expression is equivalent to the free floating Jacobian given in (6.27)*

$$J^* = J\bar{S}, \quad (6.28)$$

where

$$\bar{S} = \begin{pmatrix} -A_{bb}^{-1}A_{br} \\ I \end{pmatrix}, \quad (6.29)$$

is an alternative expression of the dynamically weighted generalized inverse of the selection matrix S shown in (6.4).

Proof. Let us first prove that (6.29) is equal to (6.4). The inverse of the block expression of A shown in (6.11) can be written in terms of Schur complements (Kailath et al. 2000) as

$$A^{-1} = \begin{pmatrix} A_{bb}^{-1} \left(I + A_{br}P^{-1}A_{rb}A_{bb}^{-1} \right) & -A_{bb}A_{br}P^{-1} \\ -P^{-1}A_{rb}A_{bb}^{-1} & P^{-1} \end{pmatrix}, \quad (6.30)$$

where

$$P \triangleq A_{rr} - A_{rb}A_{bb}^{-1}A_{br} \quad (6.31)$$

is the Schur complement of A_{bb} . As we can see

$$A^* = \left(SA^{-1}S^T \right)^{-1} = P, \quad (6.32)$$

and therefore

$$\bar{S} \triangleq A^{-1}S^T A^* = \begin{pmatrix} -A_{bb}^{-1}A_{br}P^{-1} \\ P^{-1} \end{pmatrix} P = \begin{pmatrix} -A_{bb}^{-1}A_{br} \\ I \end{pmatrix}. \quad (6.33)$$

□

Lemma 6.1.3 (Dynamically consistent generalized inverse of J^*). *The following expression is a dynamically consistent generalized inverse of J^**

$$\bar{J}^* \triangleq (A^*)^{-1}J^{*T}\Lambda^*, \quad (6.34)$$

where

$$\Lambda^* \triangleq \left(J^*(A^*)^{-1}J^{*T} \right)^{-1} \quad (6.35)$$

is a projection in task space of the inverse free floating inertia $(A^*)^{-1}$.

Proof. Because \bar{J}^* will give us the correspondence between task torques and accelerations as we will be show in (6.37) and it will also define the task's null space behavior as we will show in (6.45), we will refer to it as the dynamically consistent generalized inverse of J^* . \square

The above kinematic representations will allow us to develop operational space controllers for free space behaviors and to monitor task feasibility at runtime by studying the singular values of J^* . The singularities of J^* will be very different from the singularities of the full Jacobian because the full Jacobian reflects the contributions from both passive and active DOFs. On the other hand the free floating Jacobian reflects the dependency of the active DOFs on the passive DOFs.

6.2 Operational Space Control

Humanoids should be able to perform a variety of highly skilled tasks in mid air, such as intercepting objects, jumping, or performing acrobatic movements. Our approach to tackle these issues is once more to design whole-body controllers that can simultaneously control multiple operational tasks as well as the robot's posture while characterizing the conservation of the system's momentum.

6.2.1 Task Dynamics and Control

We consider an arbitrary task point x describing the coordinates of a desired part of the robot's body with the velocity representation given in (6.23). We can characterize the task's dynamic behavior in free space by left-multiplying (6.2) by the term JA^{-1} , yielding the following task space equation

$$\ddot{x} - \dot{J} \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + JA^{-1}(b + g) = JA^{-1}S^T\Gamma, \quad (6.36)$$

where we have used the equality $\ddot{x} = J \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} + \dot{J} \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}$.

Lemma 6.2.1 (Task dynamic behavior in free space). *The following equation of motion describes the behavior of the task in free space due to actuation torques*

$$\Lambda^* \ddot{x} + \mu^* + p^* = \bar{J}^{*T} \Gamma \quad (6.37)$$

where Λ^* is the inertial quantity defined in (6.35) and

$$\mu^* \triangleq \Lambda^* J A^{-1} b - \Lambda^* \dot{J}^* \dot{q} \quad (6.38)$$

$$p^* \triangleq \Lambda^* J A^{-1} g \quad (6.39)$$

are Coriolis/centrifugal and gravity terms.

Proof. The above equation of motion can be obtained by left-multiplying (6.36) by Λ^* , using the following alternative expression of \bar{J}^{*T}

$$\bar{J}^{*T} = \Lambda^* J A^{-1} S^T. \quad (6.40)$$

This last expression can be demonstrated by using the expression of \bar{J}^* given in (6.34), the expression of A^* given in (6.3), the expression of J^* given in (6.28), the property $\bar{S} S A^{-1} = A^{-1} S^T \bar{S}^T$ shown in (6.9), and the property of generalized inverses $S \bar{S} S = S$. \square

Theorem 6.2.1 (Operational space motion control). *The following torque vector yields linear control of task accelerations during free floating movements*

$$\Gamma = J^{*T} \left(\Lambda^* a^{ref} + \mu^* + p^* \right). \quad (6.41)$$

Here a^{ref} is a desired control policy in acceleration space.

Proof. Plugging the above expression in (6.37) and using the equality $J^* \bar{J}^* = I$ yields the desired linear behavior

$$\ddot{x} = a^{ref}. \quad (6.42)$$

\square

Residual Task Redundancy

To complete the proposed operational space formulation for free-flying behaviors let us characterize the task's residual redundancy. The task's redundant behavior can be characterized in torque space by adding an additional term with null effect on the task

Theorem 6.2.2 (Whole-body control structure in free space). *The following control structure provides linear control of task accelerations and defines the task's residual movement redundancy*

$$\Gamma = J^{*T}F + N^{*T}\Gamma_0. \quad (6.43)$$

Here,

$$F \triangleq \Lambda^* a^{ref} + \mu^* + p^* \quad (6.44)$$

is a force level control vector appearing in (6.41), N^* is the null space matrix of J^* and Γ_0 is an arbitrary torque vector acting in the null space.

Proof. While the term $J^{*T}F$ provides linear control of task accelerations as discussed in Theorem 6.2.1, the above null space term provides no coupling effects between null space torques and task space accelerations as we will show next. In turn, the vector Γ_0 can be used to control the robot's postural behavior or additional operational tasks without interfering with the primary task. \square

Corollary 6.2.1 (Dynamically consistent null space matrix in free space). *The following null space matrix projects secondary control criteria into null task accelerations*

$$N^* \triangleq I - \bar{J}^* J^*. \quad (6.45)$$

Proof. Plugging the above expression into (6.43) and the resulting expression into (6.37) yields the desired cancelation of terms. \square

6.2.2 Task Feasibility

In Chapter 2 we studied task feasibility under supporting constraints. During free floating movements the robot is not in contact with the ground, however the laws of physics impose that the robot's momentum is conserved imposing constraints in the overall movement. Therefore, task feasibility during free floating behaviors can be determined by studying the impact of momentum conservation into the task. Because we have characterized the kinematic and dynamic quantities of the robot in free space, we can measure task feasibility using the condition number of the free floating Jacobian J^* or the condition number of the inverse free floating inertia Λ^* . In contrast, task feasibility cannot be measured using the full Jacobian or the full inertia, because they do not reflect the impact of momentum conservation. In particular Λ^* will become singular when arbitrary actuation torques yield zero task accelerations, while J^* will become singular when task velocities cannot take arbitrary values.

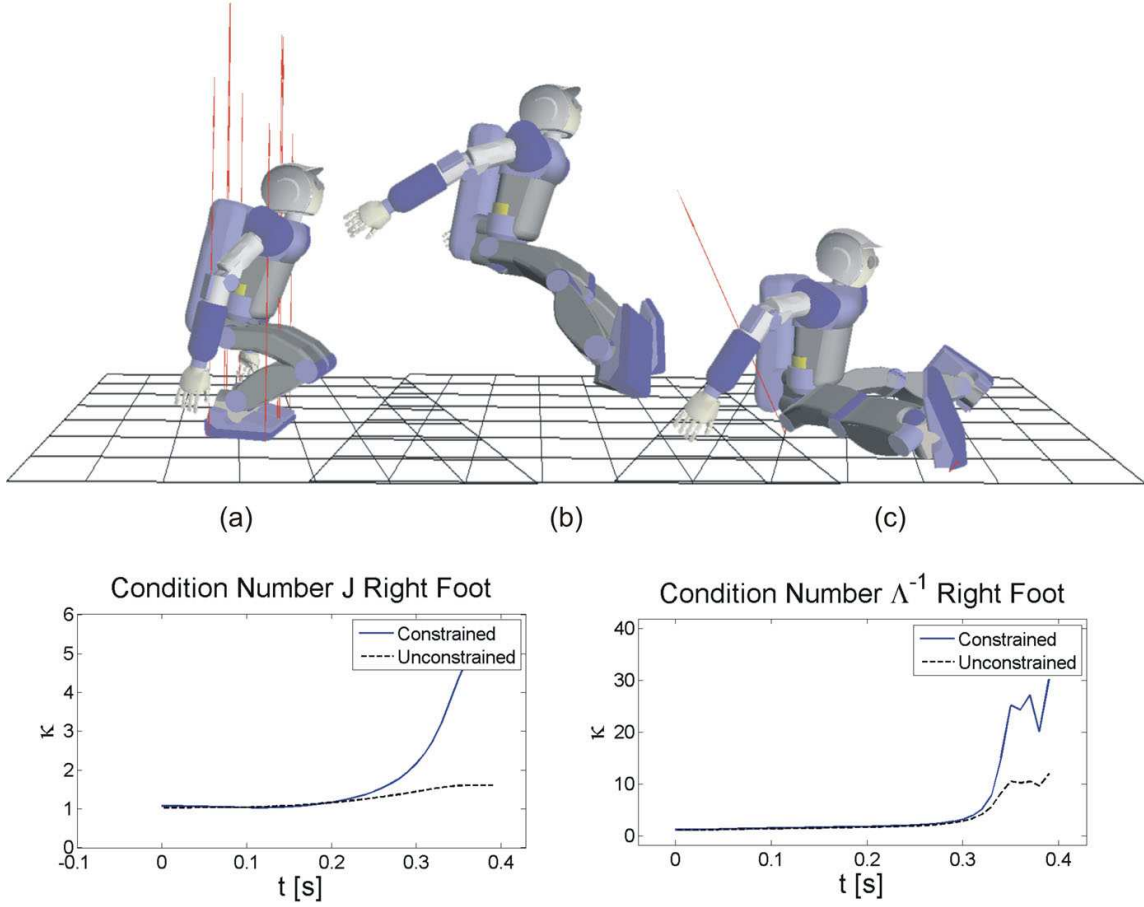


Figure 6.2: **Task feasibility in free space:** The snapshots (a), (b), and (c) depict a jumping behavior where positioning the legs forward becomes infeasible. When the robot goes in free flying mode, the feet are controlled in cartesian space to stretch forward. When the legs reach full stretch the task becomes unfeasible. The condition numbers of the free floating (constrained) Jacobian and the inverse inertia reflect the singular behavior. The condition number of the full (unconstrained) Jacobian and inverse inertia are also shown for comparison. Their value remain within bounds.

The condition numbers of the free floating Jacobian and inverse inertia for an r -dimensional task are shown below

$$\kappa(J^*) \triangleq \frac{\sigma_1(J^*)}{\sigma_r(J^*)}, \quad \kappa(\Lambda^*) \triangleq \frac{\sigma_1(\Lambda^*)}{\sigma_r(\Lambda^*)}. \quad (6.46)$$

When the task becomes infeasible these condition numbers grow to infinity, giving us valuable information to monitor task feasibility and the support to modify behavior during conflicting scenarios.

An example demonstrating task feasibility is shown in Figure 6.2. Here, a simulated jumping behavior is created and controlled using the proposed operational space control structures. When the robot loses contact with the ground the feet are commanded to stretch forward to an unreachable position. When the legs reach the maximum stretch the task becomes infeasible. Monitoring task feasibility allows us to handle conflicting scenarios at runtime.

6.3 Prioritized Multi-Task Control

As in ground based movements, free space behaviors require the simultaneous coordination of multiple operational tasks and postures. An example is shown in Figure 6.3 where a

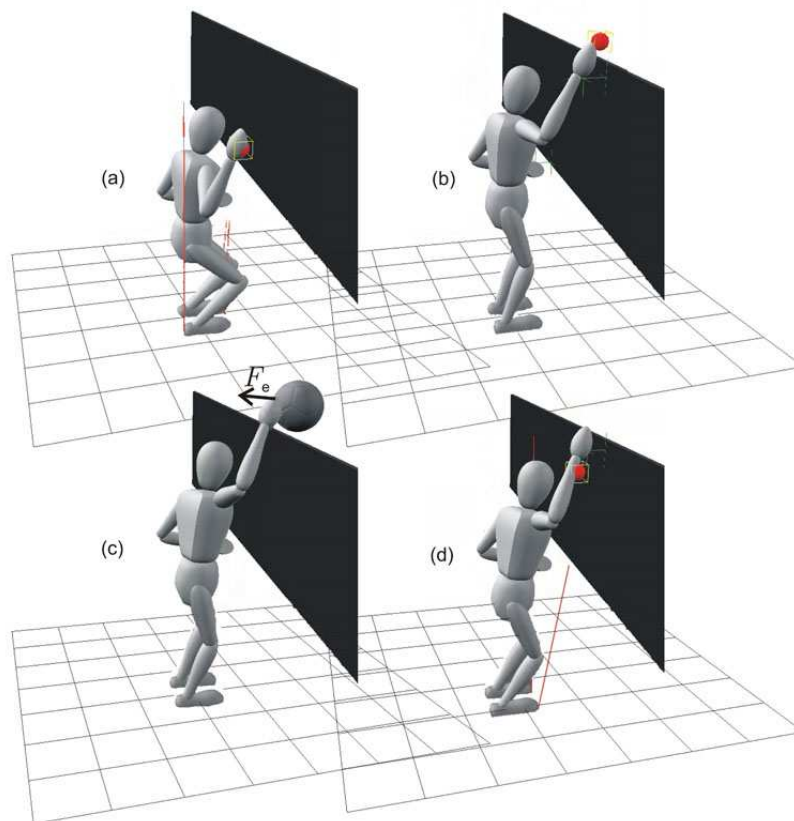


Figure 6.3: **Multi task free floating behavior:** This sequence depicts a control scenario involving hitting a ball in mid air. Multiple tasks need to be simultaneously controlled in the air to accomplish the desired action.

hitting task in mid air has been implemented controlling the tasks shown in Table 6.1. Here, feet position and orientation are controlled in preparation for landing, the robot’s right hand is controlled to intercept the ball in mid air, joint limits are monitored and if approached they locked, etc.

The goal of this section is to develop whole-body control structures that can handle multiple tasks during free flying behaviors. Handling simultaneously operational tasks, constraints, and postures will be once more our objective. To decouple high priority tasks from lower priority tasks and to monitor task feasibility we will implement a prioritized control strategy as we did in Chapter 3.

Task Decomposition (Volleyball Hit)

<i>Task Primitive</i>	<i>Coordinates</i>	<i>DOFs</i>	<i>Control Policy</i>
Joint Limits	joint positions	variable	locking attractor
Feet	position and orientation of feet	6×2 feet	position
Right Hand	position and orientation of hand	6	position
Gaze	head orientation	2 (\perp plane)	position
Posture	joint coordinates	$n = \text{NumJoints}$	optimal criterion

Table 6.1: **Volleyball task decomposition.**

6.3.1 Representations and control structures

In Table 6.2 we illustrate our choice of priorities for the previous volleyball hitting behavior. This selection is determined according to the relative importance of each task. Joint limit control is the highest priority task because it prevents damaging the robot’s body. Feet control is next because it determines landing stability. Head orientation follows because the robot’s vision system needs to track the ball before it can hit it. Hand position and orientation control has been broken down into two parts with position control taking higher priority than orientation control since the former is more relevant for hitting. The posture has the lowest priority to access the available residual redundancy.

Task Hierarchy (Volleyball Hit)

<i>Task Primitive</i>	<i>Priority Level</i>
Joint limit lock	1
Feet position and orientation	2
Head orientation	3
Hand position	4
Hand orientation	5
Posture (captured pose)	6

Table 6.2: **Task hierarchy.****Constrained Kinematics**

The full kinematic representation of an arbitrary task point k is

$$x_k = \begin{pmatrix} x_{k,p} \\ x_{k,r} \end{pmatrix}, \quad (6.47)$$

where $x_{k,p}$ is a position representation of the task point and $x_{k,r}$ is an orientation representation. The instantaneous kinematics of arbitrary task points is expressed in terms of base and joint velocities as

$$\dot{x}_k = J_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (6.48)$$

where J_k is the task Jacobian in task coordinates and ϑ_b and \dot{q} are base and joint velocities respectively. Similarly to (6.26) we can represent arbitrary task velocities in free space as

$$\dot{x}_k = J_k^* \dot{q} + \dot{x}_{k,\text{bias}}, \quad (6.49)$$

where

$$J_k^* \triangleq J_{k,r} - V_{k,b} A_{bb}^{-1} A_{br}. \quad (6.50)$$

Here $x_{k,\text{bias}}$ is a generalization of the bias term given in (6.26) and $J_k = \begin{pmatrix} V_{k,b} & J_{k,r} \end{pmatrix}$ is a generalization of (6.24).

Definition 6.3.1 (Free floating Jacobian of arbitrary points). *The following kinematic representation is a generalization of the free floating Jacobian shown in (6.28)*

$$J_k^* = J_k \bar{S}. \quad (6.51)$$

Prioritization

Like in Chapter 3, we will develop here control structures based on null space projections. Similarly to (3.11) we propose to use the following prioritized control structure to control multiple tasks

$$\Gamma = \Gamma_1 + \Gamma_{2|\text{prec}(2)} + \cdots + \Gamma_{N|\text{prec}(N)} = \sum_{k=1}^N \Gamma_{k|\text{prec}(k)}, \quad (6.52)$$

where the subscript $k|\text{prec}(k)$ is used to indicate that the k -th task operates in the null space of all higher priority tasks.

Definition 6.3.2 (Prioritized torques). *The following expression determines the projection of lower priority tasks into the null space of higher priority tasks*

$$\Gamma_{k|\text{prec}(k)} \triangleq N_{\text{prec}(k)}^T \Gamma_k, \quad (6.53)$$

where $N_{\text{prec}(k)}$ is the combined null space of all higher priority tasks (i.e. all preceding tasks) to the k -th level.

Once more, we will be able to formulate a general operational space control structure that will take the form

$$\Gamma = \left(J_1^{*T} F_1 \right) + \left(J_{2|1}^{*T} F_{2|1} \right) + \cdots + \left(J_{N|\text{prec}(N)}^{*T} F_{N|\text{prec}(N)} \right), \quad (6.54)$$

where the matrices $J_{k|\text{prec}(k)}^*$ correspond to prioritized task Jacobians as will be defined in (6.55), and the vectors $F_{k|\text{prec}(k)}$ correspond to control forces to control the k -th priority task.

Definition 6.3.3 (Prioritized Jacobian). *The following prioritized Jacobian is associated with the k -th priority task*

$$J_{k|\text{prec}(k)}^* \triangleq J_k^* N_{\text{prec}(k)}^*, \quad (6.55)$$

where J_k^* is the constrained Jacobian associated with the k -th task as shown in (6.51) and $N_{\text{prec}(k)}^*$ is the prioritizing null space of all preceding tasks and will be characterized in (6.69).

Task Dynamics and Control

We can derived task dynamics by left-multiplying (6.2) by the term $J_k A^{-1}$, where J_k is the full Jacobian of the k -th operational task yielding the following equation of motion

$$\ddot{x}_k - \dot{J}_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + J_k A^{-1}(b + g) = J_k A^{-1} S^T \left(\Gamma_{k|\text{prec}(k)} + \sum_{(i>0) \wedge (i \neq k)}^N \Gamma_{i|\text{prec}(i)} \right). \quad (6.56)$$

Here we have used the equality

$$\ddot{x}_k = J_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + \dot{J}_k \begin{pmatrix} \dot{\vartheta}_b \\ \ddot{q} \end{pmatrix} \quad (6.57)$$

and we have decomposed the actuation torques into torques allocated to control the k -th operational task and torques allocated to control all other operational tasks, i.e.

$$\sum_{i=1}^N \Gamma_{i|\text{prec}(i)} = \left(\Gamma_{k|\text{prec}(k)} + \sum_{(i>0) \wedge (i \neq k)}^N \Gamma_{i|\text{prec}(i)} \right). \quad (6.58)$$

Definition 6.3.4 (Prioritized inertia). *The following term is referred to as the prioritized inertia of the k -th priority task*

$$\Lambda_{k|\text{prec}(k)}^* \triangleq \left(J_{k|\text{prec}(k)}^* (A^*)^{-1} J_{k|\text{prec}(k)}^{*T} \right)^{-1}. \quad (6.59)$$

Theorem 6.3.1 (Prioritized operational space motion control). *The following control vector yields linear control of accelerations for the k -th prioritized task*

$$\Gamma_{k|\text{prec}(k)} = J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)}, \quad (6.60)$$

where $\Gamma_{k|\text{prec}(k)}$ is the k -th component of the prioritized torque control structure shown in (6.52), $J_{k|\text{prec}(k)}^*$ is the prioritized Jacobian for the k -th task point discussed in (6.55), and $F_{k|\text{prec}(k)}$ is a vector of control forces that will be discussed in a few lines.

Proof. Based on the above control term, (6.56) becomes

$$\ddot{x}_k - \dot{J}_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + J_k A^{-1}(b + g) = \left(\Lambda_{k|\text{prec}(k)}^* \right)^{-1} F_{k|\text{prec}(k)} + J_k A^{-1} S^T \sum_{(i>0) \wedge (i \neq k)}^N \Gamma_{i|\text{prec}(i)}, \quad (6.61)$$

where the term $\Lambda_{k|\text{prec}(k)}^*$ is the inertial term defined in (6.59) whose inverse maps prioritized forces into task accelerations and fulfill the following equality which can be demonstrated in a similar way than we did in Property 3.2.1

$$\left(\Lambda_{k|\text{prec}(k)}^* \right)^{-1} = J_k A^{-1} S^T J_{k|\text{prec}(k)}^{*T}. \quad (6.62)$$

Under normal conditions $\Lambda_{k|\text{prec}(k)}^*$ is full rank, and therefore the vector $F_{k|\text{prec}(k)}$ yields linear control of task accelerations and forces. \square

Corollary 6.3.1 (Prioritized motion control). *The following control vector yields linear control of task accelerations*

$$F_{k|\text{prec}(k)} \triangleq \Lambda_{k|\text{prec}(k)}^* a_k^{ref} + \mu_{k|\text{prec}(k)}^* + p_{k|\text{prec}(k)}^* - \Lambda_{k|\text{prec}(k)}^* J_k A^{-1} S^T \sum_{i=1}^{k-1} \Gamma_{i|\text{prec}(i)}. \quad (6.63)$$

Here $F_{k|\text{prec}(k)}$ is the control force shown in (6.60), a_k^{ref} is an acceleration-level control policy for the k -th priority task, and the remaining dynamic quantities for the above equation have the following expressions,

$$\mu_{k|\text{prec}(k)}^* \triangleq \Lambda_{k|\text{prec}(k)}^* J_k A^{-1} b - \Lambda_{k|\text{prec}(k)}^* \dot{J}_k \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix}, \quad (6.64)$$

$$p_{k|\text{prec}(k)}^* \triangleq \Lambda_{k|\text{prec}(k)}^* J_k A^{-1} g. \quad (6.65)$$

Proof. It is straightforward to verify that using the above expressions in (6.61) will yield the linear behavior

$$\ddot{x}_k = a_k^{ref}. \quad (6.66)$$

\square

Corollary 6.3.2 (Prioritized multi-task control structure). *The following control structure yields linear control of accelerations of a set of N prioritized tasks*

$$\Gamma = \left(J_1^{*T} F_1 \right) + \left(J_{2|1}^{*T} F_{2|1} \right) + \cdots + \left(J_{N|\text{prec}(N)}^{*T} F_{N|\text{prec}(N)} \right) = \sum_{k=1}^N J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)}. \quad (6.67)$$

Proof. Using (6.60) we can further express (6.52) as the above aggregation of prioritized operational space control structures. \square

Recursive Redundancy

Null space projections impose that lower priority tasks do not introduce acceleration and force components in higher priority tasks. Given Equation (6.56) this is equivalent to the following condition

$$\forall i \in \text{prec}(k) \quad J_i A^{-1} S^T N_{\text{prec}(k)}^{*T} = 0. \quad (6.68)$$

Similarly than we did in Corollary 3.2.5 and without going into detail the following corollary defines the recursive null space expression

Corollary 6.3.3 (Compact expression of $N_{\text{prec}(k)}^*$). *The null space matrix that fulfills the above set of constraints can be expressed using the following compact expression*

$$N_{\text{prec}(k)}^* = I - \sum_{i=1}^{k-1} \bar{J}_{i|\text{prec}(i)}^* J_{i|\text{prec}(i)}^*, \quad (6.69)$$

Proof. The proof for this expression is analogous to the proofs used for Corollaries 3.2.3 and 3.2.5. \square

Corollary 6.3.4 (Dynamically consistent generalized inverse of $J_{k|\text{prec}(k)}^*$). *The following expression is the dynamically consistent generalized inverse of $J_{k|\text{prec}(k)}^*$*

$$\bar{J}_{k|\text{prec}(k)}^* \triangleq (A^*)^{-1} J_{k|\text{prec}(k)}^{*T} \Lambda_{k|\text{prec}(k)}^*. \quad (6.70)$$

Proof. The proof for the above expression is analogous to that given for Corollary 3.2.4. \square

6.3.2 Task Feasibility

The constrained Jacobian $J_{k|\text{prec}(k)}^*$ given in (6.55) or the constrained inertia matrix $\Lambda_{k|\text{prec}(k)}^*$ given in (6.59), reflect the impact of momenta conservation and prioritization and can be

used to determine the feasibility of the operating tasks. To study task feasibility we study their condition numbers, i.e.

$$\kappa\left(J_{k|\text{prec}(k)}^*\right) \triangleq \frac{\sigma_1\left(J_{k|\text{prec}(k)}^*\right)}{\sigma_r\left(J_{k|\text{prec}(k)}^*\right)}, \quad \kappa\left(\Lambda_{k|\text{prec}(k)}^*\right) \triangleq \frac{\sigma_1\left(\Lambda_{k|\text{prec}(k)}^*\right)}{\sigma_r\left(\Lambda_{k|\text{prec}(k)}^*\right)}, \quad (6.71)$$

where $\kappa(\cdot)$ represents the condition number and $\sigma_i(\cdot)$ represents the i -th singular value of the enclosed matrix.

6.3.3 Posture Dynamics and Control

During free flying behaviors the posture plays an important role in determining the overall movement behavior. For instance, the posture determines landing stability or the optimal orientation of the body during jumping and running behaviors. Posture control structures for free space will be very similar to those of Chapter 4. In fact, whole-body control is characterized once more using the prioritized expression given in Once more, the whole-body control structure for free space movements will have the same form than in Theorem 4.1.1.

Theorem 6.3.2 (Whole-body prioritized control structure). *The following control structure for free space behaviors provides linear control of N prioritized tasks and defines the postural space of motion*

$$\Gamma = \sum_{k=1}^N J_{k|\text{prec}(k)}^{*T} F_{k|\text{prec}(k)} + N_t^{*T} \Gamma_{\text{posture}}, \quad (6.72)$$

where,

$$N_t^* \triangleq I - \sum_{k=1}^N \bar{J}_{k|\text{prec}(k)}^* J_{k|\text{prec}(k)}^*, \quad (6.73)$$

is the null-space of all prioritized tasks and can be derived using the recursive expression presented in (6.69). Here, N_t^* signifies the null space of all priority tasks.

Proof. The proof is analogous to that of Theorem 4.1.1. □

Once more, we will use the more compact notation

$$\Gamma = \sum_{k=1}^N \Gamma_{k|\text{prec}(k)} + \Gamma_{p|t} = \Gamma_t + \Gamma_{p|t}, \quad (6.74)$$

where

$$\Gamma_{p|t} \triangleq N_t^{*T} \Gamma_{\text{posture}} \quad (6.75)$$

are torques operating in the task's residual redundancy and will be used to control the robot's posture.

Task Based Postures

Here the objective is to control the position of postural DOFs as we described in Section 4.2 of Chapter 4. Postural dynamic behavior can be obtained by left-multiplying (6.2) by the term $J_p A^{-1}$ where J_p is the full Jacobian of the posture,

$$\ddot{x}_p - \dot{J}_p \begin{pmatrix} \vartheta_b \\ \dot{q} \end{pmatrix} + J_p A^{-1} (b + g) = J_p A^{-1} S^T \left(\Gamma_{p|t} + \sum_{k=1}^N \Gamma_{k|\text{prec}(k)} \right), \quad (6.76)$$

where x_p is a set of postural DOFs and J_p is the associated Jacobian and we have used the torque decomposition given in (6.74).

Proposition 6.3.1 (Posture motion control). *The following torque vector for free space behaviors yields linear control of postural accelerations*

$$\Gamma_{p|t} = J_{p|t}^{*T} \left(\Lambda_{p|t}^* a_p^{ref} + \mu_{p|t}^* + p_{p|t}^* - \Lambda_{p|t}^* J_p A^{-1} S^T \sum_{k=1}^N \Gamma_{k|\text{prec}(k)} \right), \quad (6.77)$$

where a_p^{ref} is a joint feedback control policy for all postural DOFs, $J_{p|t}^* \triangleq J_p^* N_t^*$ is the prioritized posture Jacobian discussed in (6.55),

$$\Lambda_{p|t}^* = \left(J_{p|t}^* (A^*)^{-1} J_{p|t}^{*T} \right)^{-1}, \quad (6.78)$$

is a posture inertial term similar to (6.59), and $\mu_{p|t}^*$ and $p_{p|t}^*$ are Coriolis/centrifugal and gravity terms with similar expressions than (6.64) and (6.65).

Proof. Because this controller is based on Corollary 6.3.1, it will yield the linear behavior

$$\ddot{x}_{\text{posture}} = a_p^{ref}. \quad (6.79)$$

□

Criteria based postures

To optimize postural criteria we first characterized the dynamic behavior in joint space. Here, the posture coordinates are equal to all joint coordinates. For a set q_p of postural coordinates as described in Definition 4.3.2, posture dynamics can be obtained by left-multiplying (6.2) by the term $S_p A^{-1}$, where S_p is the posture selection matrix, yielding the following joint space equation of motion

$$\ddot{q}_p + S_p A^{-1}(b + g) = S_p A^{-1} S^T \left(\Gamma_{p|t|c} + \sum_{k=1}^N \Gamma_{k|\text{prec}(k)} \right). \quad (6.80)$$

Definition 6.3.5 (Prioritized postural Jacobian). *The following matrix represents the prioritized postural Jacobian with similar derivation than (6.55)*

$$S_{p|t}^* = S_p^* N_t^*, \quad (6.81)$$

where N_t^* is the null space matrix shown in (6.73) and

$$S_p^* = S_p \bar{S} \quad (6.82)$$

is the constrained postural Jacobian with similar derivation than (2.47).

Definition 6.3.6 (Prioritized inverse postural inertia). *The following expression is referred to as the prioritized inverse posture inertia*

$$\Phi_{p|t}^* \triangleq S_{p|t}^* (A^*)^{-1} S_{p|t}^{*T}. \quad (6.83)$$

In general the above inverse inertia will not be full rank because q_p is normally larger than the number of available DOFs within the tasks's residual redundancy.

Theorem 6.3.3 (Postural criteria-based control). *The following posture control vector for free space behaviors will yield optimal gradient descent of postural criteria*

$$\Gamma_{p|t} = S_{p|t}^{*T} F_{p|t}, \quad (6.84)$$

where $F_{p|t}$ is a posture control vector with the following expression

$$F_{p|t} = \left(\Phi_{p|t}^* \right)^+ \alpha_p^{ref} + b_{p|t}^* + g_{p|t}^* - \left(\Phi_{p|t}^* \right)^+ S_p A^{-1} S^T \Gamma_t. \quad (6.85)$$

Here $(.)^+$ is the Moore-Penrose pseudo-inverse, α_p^{ref} is a control policy implementing gradient descent of postural criteria, and the following vectors are Coriolis/centrifugal and gravity terms

$$b_{p|t}^* \triangleq (\Phi_{p|t}^*)^+ S_p A^{-1} b, \quad (6.86)$$

$$g_{p|t}^* \triangleq (\Phi_{p|t}^*)^+ S_p A^{-1} g. \quad (6.87)$$

Proof. The proof for the above theorem is analogous to the proof for Theorem 4.3.1 of Chapter 4 and therefore will yield the desired linear behavior in the controllable posture directions, i.e.

$$U_r^T (\ddot{q}_p = \alpha_p^{ref}), \quad (6.88)$$

where U_r is the basis of controllable directions in posture space and are determined through the following eigen-decomposition

$$(\Phi_{p|t}^*)^+ = U_r \Sigma_r^{-1} U_r^T. \quad (6.89)$$

□

Proposition 6.3.2 (Gradient descent control law). *The following PD control law implements postural gradient descent while providing a velocity saturation mechanism*

$$\alpha_p^{ref} = -k_v (\dot{q}_p - \nu_v \omega_{des}) \quad (6.90)$$

$$\omega_{des} = -\frac{k_p}{k_v} \nabla V_p, \quad \nu_v = \min\left(1, \frac{\omega_{max}}{\|\omega_{des}\|}\right). \quad (6.91)$$

where ν_v is the saturation term, ω_{max} is the maximum allowable angular velocity, ∇V_p is the gradient of V_p with respect to q_p , and k_p and k_v are proportional and differential gains respectively.

Proof. The proof of the above proposition is analogous to that of Proposition 4.3.1. □

6.4 Examples

6.4.1 Forward Jumping

We study the jumping behavior shown in Figure 6.4. To create this behavior we sequence a collection of states corresponding to whole-body movements. Sequencing of movements is a topic that will be discussed in the next chapter. A jumping behavior results from sequencing

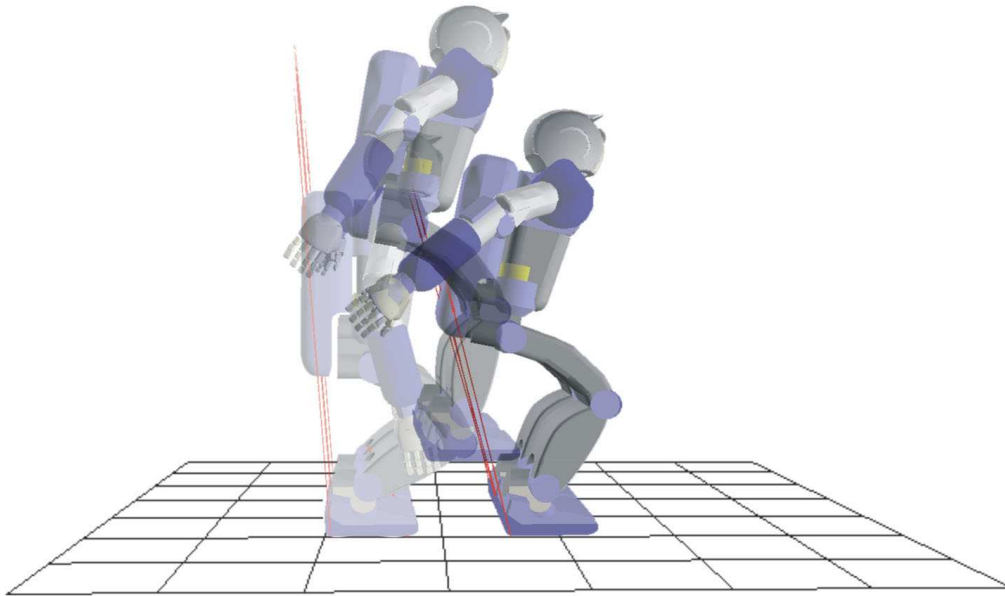


Figure 6.4: **Forward jumping behavior:** These superimposed snapshots are taken from an experiment on jumping. During the free flying phase, the whole-body controller proposed in this chapter is implemented.

ground level movements and free space movements. When the robot is in contact with the ground, reaction forces are used to provide balance stability and to push the body upwards. To control ground-based movements we use the whole-body control structure given in (5.30) of the previous chapter. The task decomposition associated with ground phases is shown in the following table,

Task Decomposition (Vertical Jump): Ground-Based Phases

<i>Task Primitive</i>	<i>Coordinates</i>	<i>DOFs</i>	<i>Priority</i>	<i>Control Policy</i>
Joint Limits	joint positions	variable	1	locking attractor
COG Vertical	COG(z)	1	2	trajectory
COG Horizontal	COG(x, y)	2	3	position
Head	head orientation	2 (\perp plane)	4	position
Upright posture	hip orientation	2	5	position
Joint posture	joint coordinates	$n = \text{NumJoints}$	6	optimal criterion

This set of tasks is used to create movements involving crouching down and accelerating the body upwards. To move the robot to a crouching pose the goal of the COG's vertical position is set to a predetermined value close to the ground, in our case $0.5m$. To jump forward, we command the COG to track an inclined trajectory at a speed equal to $1.7 m/s$ with an elevation angle equal to 75° with respect to the ground plane. An upright posture is implemented to keep the upper body vertical. On the other hand, in the free floating phase we implement the following set of tasks

Task Decomposition (Vertical Jump): Free Floating Phases

<i>Task Primitive</i>	<i>Coordinates</i>	<i>DOFs</i>	<i>Priority</i>	<i>Control Policy</i>
Feet	pos/ori	6×2 feet	1	position
Head	head orientation	2 (\perp plane)	2	position
Upright posture	hip orientation	2	3	position
Joint posture	joint coordinates	$n = \text{NumJoints}$	4	optimal criterion

Here the main task is to maintain the orientation of the feet horizontal to the ground and to move the feet forward to gain stability upon landing. Although the position of the feet

should depend on the estimated reaction forces upon landing and on the COG's trajectory in midair, here we choose them empirically. The above set of tasks is executed using the whole-body control structure for free space movements described in (6.72).

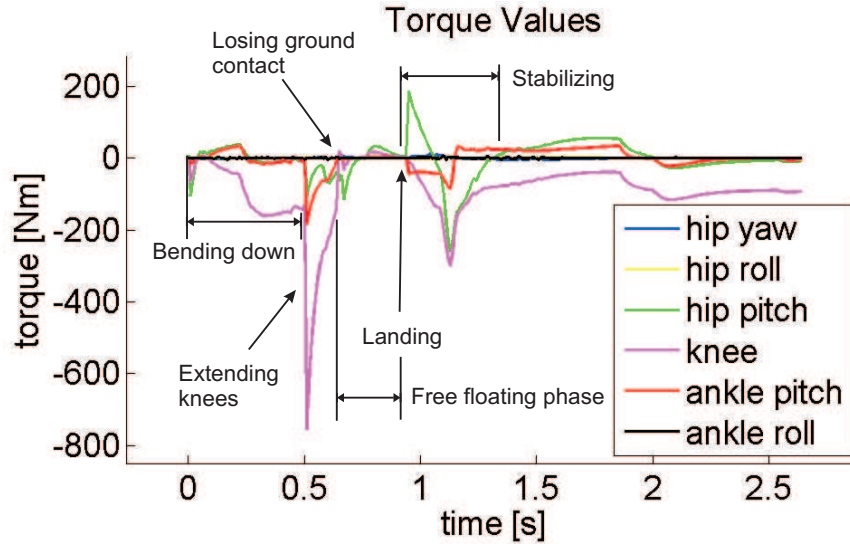


Figure 6.5: **Torques values during jumping:** Torque values corresponding to the robot's right leg during forward jumping.

The overall jumping behavior occurs by sequencing five different phases: (1) crouching down, (2) accelerating upwards, (3) releasing the body into the air (4) preparing for landing, and (5) gaining stability upon landing. Phases (1), (2) and (5) are ground based phases and phases (3) and (4) are free floating phases. The events that trigger each phase are the following. Phase (2) starts when the crouching down position is reached. Phase (3) is triggered when the feet lose ground contact. Phase (4) is triggered when the body starts to fall towards the ground. Phase (5) starts when the feet touch down. The resulting jumping height (i.e. the COG's maximum height) achieved using the previous sequence and COG trajectory is 0.35 m above the ground while the resulting jumping length is approximately 0.5 m .

Torque values during the above jumping behavior are shown in Figure 6.5. The highest torque values occur on the knees reaching peak values of more than 700 Nm . Peak values for the ankle pitch joints are 200 Nm corresponding to the period when the body accelerates upwards. The hip's pitch joint reaches peak torques around 200 Nm upon landing.

6.4.2 Kick in Mid Air

A similar jumping behavior is shown in Figure 6.6, but this time the robot's right foot is commanded to kick forward in midair. This action can be achieved by sequencing the same movements than in forward jumping but commanding the robot's right foot to move forward $0.2m$ in mid air.

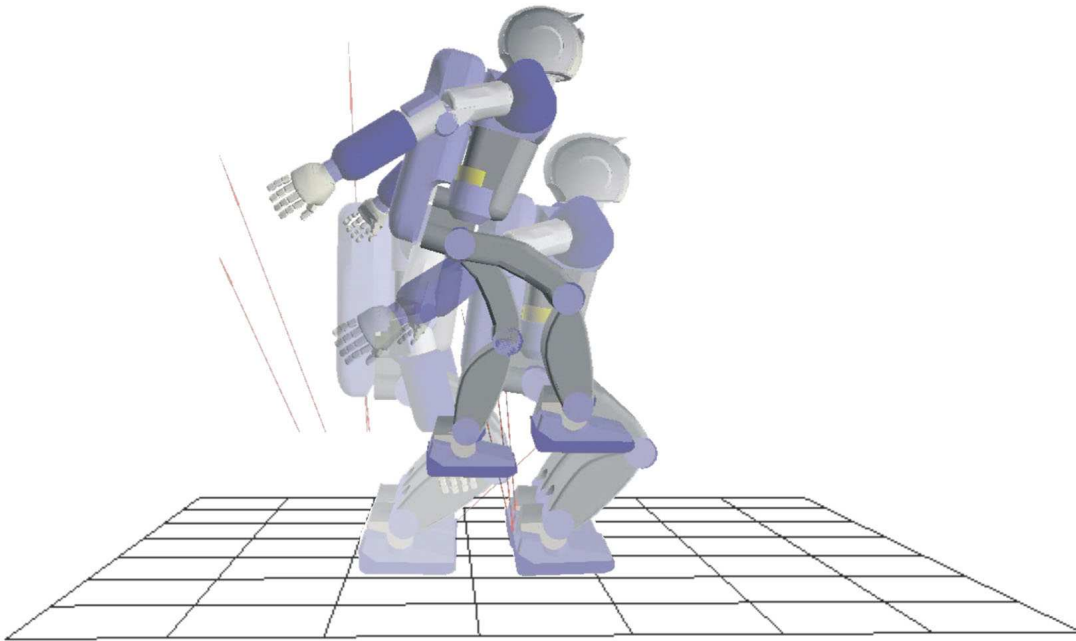


Figure 6.6: **Jumping forward with kick:** This forward jumping behavior involves kicking in midair. The foot's kicking position is provided at runtime with no previous trajectories precomputed.

6.4.3 Twist'N'Jump

A third jumping behavior is shown in Figure 6.7, where the robot's upper body is commanded to spin horizontally to create a twisting reaction moment. While accelerating the body upwards, the chest's yaw joint is accelerated counterclockwise. The set of tasks to twist the chest during the phase corresponding to accelerate the body upwards is

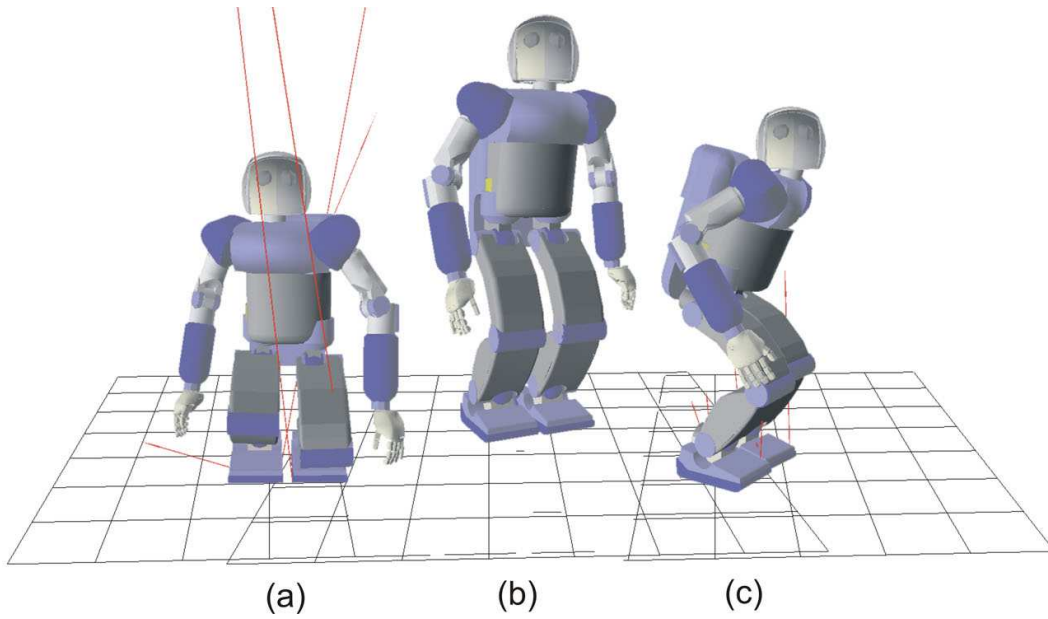


Figure 6.7: **Twist 'n' jump behavior:** This behavior is similar to a vertical jump but with an additional spin of the chest to create a twisting movement.

Task Decomposition (Twist 'N' Jump): Accelerating Body Upwards

<i>Task Primitive</i>	<i>Coordinates</i>	<i>DOFs</i>	<i>Priority</i>	<i>Control Policy</i>
Joint Limits	joint positions	variable	1	locking attractor
COG Vertical	COG(z)	1	2	trajectory
COG Horizontal	COG(x, y)	2	3	position
Head	head orientation	2 (\perp plane)	4	position
Upright posture	hip ori	2	5	position
Twist posture	chest joint	1	6	position
Joint posture	joint coordinates	$n = \text{NumJoints}$	7	captured pose

Chapter 7

Realtime Synthesis of Whole-Body Behaviors

In this chapter we will develop tools for the composition and sequencing of whole-body behaviors, representing a first step towards the design of a high-level behavioral platform for humanoid systems.

A great deal of work has been focused on the areas of behavior-based control (Brooks et al. 2004) and motion planning (Latombe 1999), addressing a variety of mobile robotic platforms. However, research on humanoid systems has been mostly focused on low-level controllers (Sentis and Khatib 2006; Kajita et al. 2003a; Fujimoto and Kawamura 1996), in part due to the difficulty of creating complex behaviors while maintaining balance stability and while complying with joint limit and self collision constraints. The control methods presented throughout this dissertation open new opportunities to create emergent behaviors in humanoid systems. In particular, our methods are capable of executing arbitrary control objectives (both at the contact and non-contact levels) while maintaining balance stability and while responding to dynamic events. The next logical step is to connect our controllers to perception and decision systems. However, a direct connection between execution and behavioral layers is not obvious. The goal of this chapter is to develop behavioral entities to interface these two layers.

While the main task of an execution layer is to create torque commands to accomplish desired control objectives, the task of a behavioral layer is to sense the environment and coordinate sets of actions to create desired responses. Much work has been focused on the synthesis of autonomous behaviors. In (Brooks et al. 2004) the synthesis of emergent behaviors as the coordinated action of distributed processes is addressed. In (Arkin 1998)

motor schemas are proposed as the basic units of action concurrently operating to control the overall behavioral response. The goal of this chapter is to develop novel control entities to support the implementation of emergent behaviors in humanoid systems.

At the execution level, our operating engine is the whole-body control framework described in previous chapters. To support the creation of low-level tasks we will create new software abstractions called control primitives. Control primitives are entities that define task representations and control policies. However, control primitives are not based on pre-defined goals or trajectories. Instead, task goals can be provided at runtime as part of the emergent behavior.

To support the creation of whole-body behaviors we will define new entities called whole-body behaviors which consist on sequences of actions accomplished by executing sets of control objectives. For instance, a whole-body behavior designed to sit down on a chair will involve placing the hands on the arms for support, lowering the hip to make contact with the seat, and resting the back on the seat support, where each of these actions is defined as a unique set of objectives.

Whole-body behaviors are therefore self-coordinated modules that can execute a specific action given the appropriate goals. In Figure 7.1 we show a control diagram involving execution and behavioral layers connected through an action layer which embodies a library of whole-body behaviors.

While research on autonomous navigation for mobile platforms and legged robots has made impressive progress in recent times, tackling high dimensional problems such as the realtime generation of manipulation and locomotion behaviors in humanoids systems is an unsolved problem. Not only the system's state is very large but controllers have to cope with arbitrary supporting contacts, balance stability, and internal constraints. To tackle this problem the action layer we will discuss here will be aimed at reducing the dimensionality of the control problem as well as at handling arbitrary contacts and internal constraints without the necessity to plan the global task. For this layer we will define action units called whole-body behaviors responsible for characterizing task decomposition and movement sequencing. Therefore, the action layer will embody a library of whole-body behaviors that can be instantiated and coordinated by a high level controller. As an example, walking or pushing behaviors would be defined as part of the action layer.

This chapter is organized as follows. In Section 7.1 we will describe the composition and instantiation of control primitives for the creation of low-level tasks. In Section 7.2 we will describe the mechanisms of the action layer in the form of whole-body behaviors.

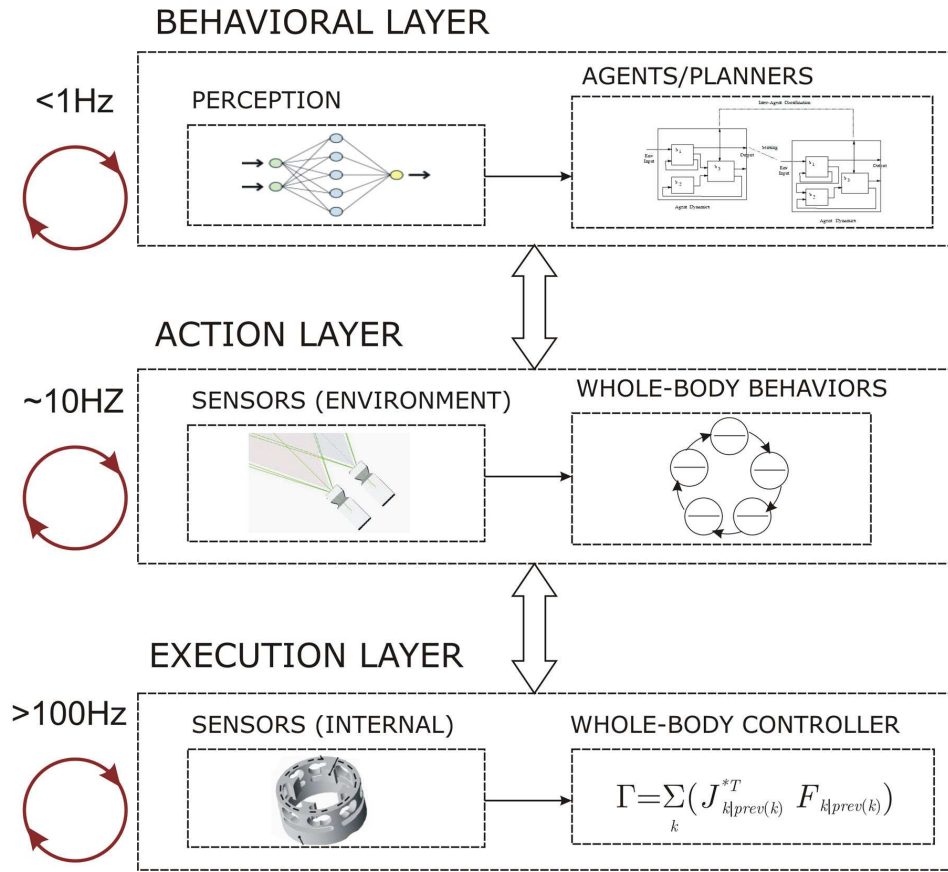


Figure 7.1: **Connection between behavior and execution layers:** The execution layer operates at fast rates with the objective of executing low-level tasks. The behavioral layer operates at low speeds sensing the environment and determining the whole-body behaviors necessary to accomplish a global task. To interface these two layers, we define an action layer which is responsible for providing access to whole-body behavioral entities. These entities are meant to serve as the main units of action orchestrated by the behavioral layer.

7.1 Composition and Instantiation of Low-Level Behaviors

To facilitate the creation of whole-body behaviors it is our objective here to provide supporting entities to create and instantiate low-level tasks as part of the composition of whole-body behaviors. These entities represent a direct mapping between desired objectives and motor commands.

In Figure 7.2 we depict our implementation of the execution layer. The centerpiece of this diagram is the whole-body controller described in previous chapters which has been designed to execute sets of tasks and to monitor task feasibility. To create new tasks, we

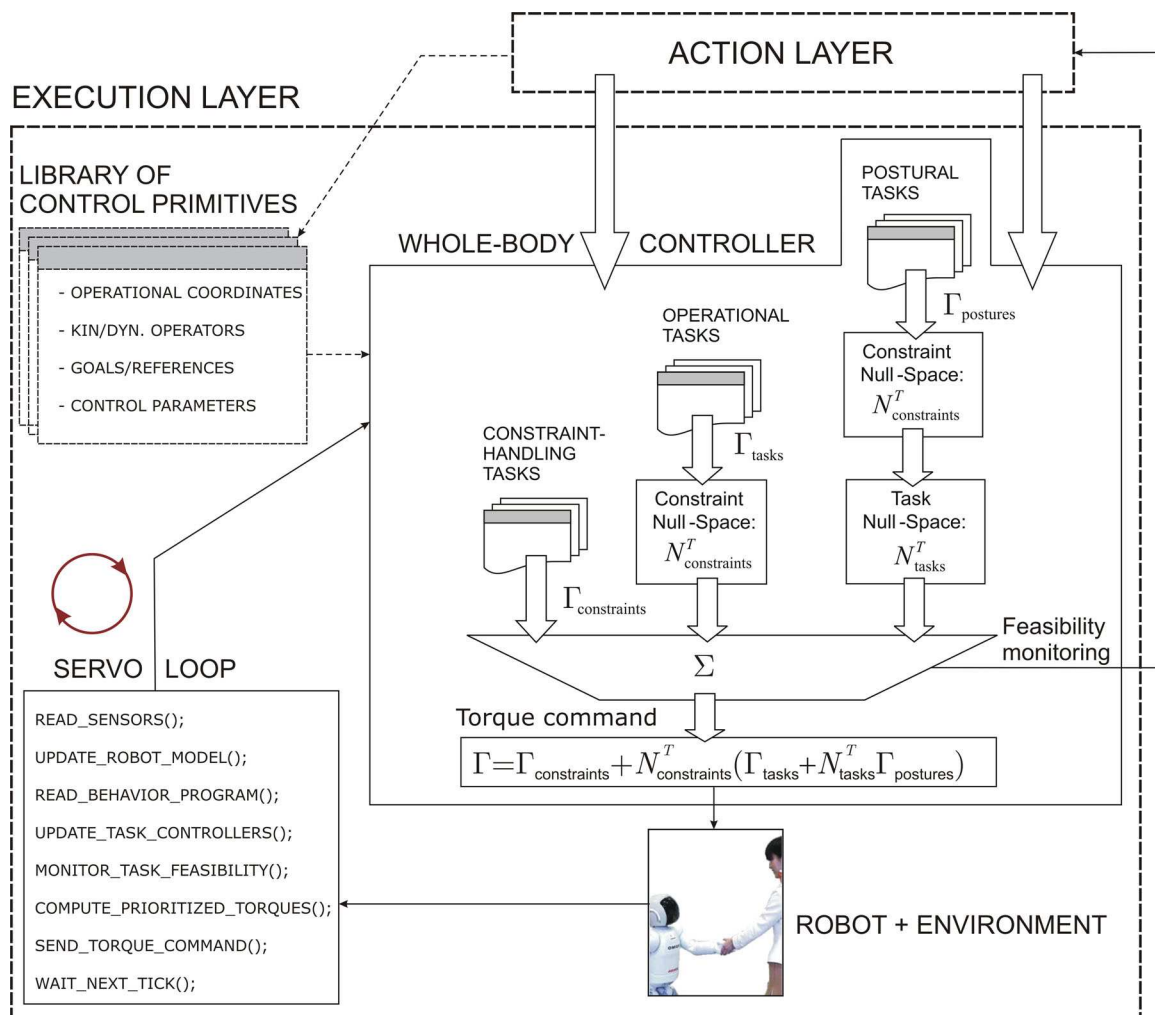


Figure 7.2: **Execution layer:** The centerpiece of this layer is the whole-body controller described in previous chapters. The definition and instantiation of low-level behaviors is supported by abstract entities called control primitives.

define control primitives that characterize task representation and control policies. The instantiation of control primitives leads to the creation of low-level behaviors associated with specific body parts (i.e. task points).

7.1.1 Control Primitives

Control primitives are software abstractions that support the creation of low-level tasks. These abstractions encapsulate task representation and control policies, serving as basic units of control. To support the implementation of control primitives we define software

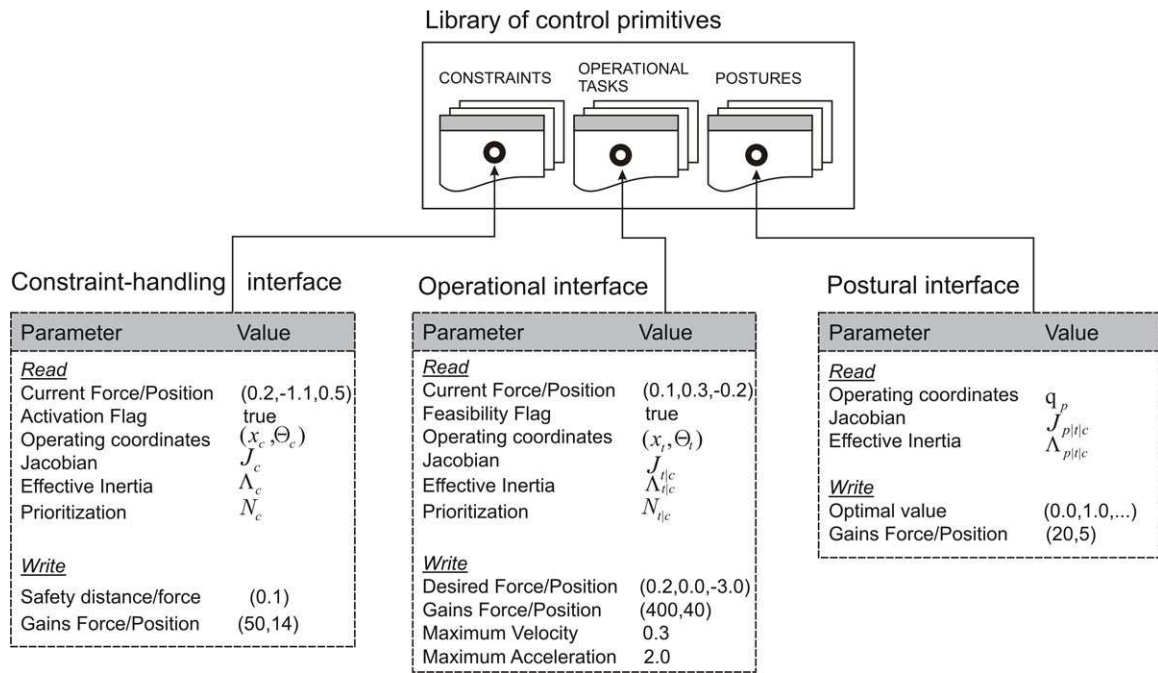


Figure 7.3: **Control primitives:** This figure shows software interfaces designed to create a variety of control primitives.

interfaces associated with the different types of tasks, as shown in Figure 7.3. Three interfaces are shown here: an interface to implement constraint-handling tasks, an interface to implement operational tasks, and an interface to implement postural tasks.

Control primitives are created as part of the description of whole-body behaviors, encapsulating the representation and functionality of different body parts. However, specific goals and control parameters do not need to be pre-programmed. Instead this information can be passed by the sensory layer at runtime. Using similar modules we have built an extensive library of control primitives to address the control of different body parts. Some of them are shown in Table 7.1.

7.1.2 Task Creation

Task creation is the process of instantiating control primitives and assigning control parameters. In Figure 7.4 we illustrate a whole-body multi-contact behavior and the task structures associated with it.

The instantiation of low-level tasks involves creating task objects and passing associated control parameters. For instance, to instantiate a hand position control task we execute the

Library of control primitives and their function

<i>Control primitive</i>	<i>Function</i>	<i>Category</i>
Joint limits avoidance	Locks violating joints	Constraint
Obstacle avoidance	Keeps safety distance	Constraint
Static balance	Controls COG position	Balance
Dynamic balance	Controls ZMP position	Balance
Position control	Controls arbitrary task positions	Operational Task
Orientation control	Controls arbitrary task orientations	Operational Task
Hybrid force/position control	Controls arbitrary force/position	Operational Task
Imitate captured pose	Imitates captured poses	Posture
Minimize effort	Minimizes torque effort	Posture

Table 7.1: **Library of control primitives:** In this table we list some control primitives we have created to support the creation of whole-body behaviors.

following C++ statements

```
PositionPrimitive* handTask;
handTask = new PositionPrimitive( robotModel, "right-hand");
```

Here, `PositionPrimitive` is an abstraction that encapsulates position representations of arbitrary parts and PD control policies of desired position commands. A class structure is associated with this primitive, containing a constructor that takes as input the robot model and the desired body part to be controlled. The robot model is characterized by the UML diagram shown in Figure 7.5.

After instantiating a task, the next step is to pass desired control parameters. For instance, to use the PD control law with velocity and acceleration saturation described in Equations (3.47) and (3.48) we pass the following parameters: $k_p = 400s^{-2}$ for the control gain, $\nu_v = 0.5m/s$ for the velocity saturation value, and $\nu_a = 3m/s^2$ for the acceleration saturation value. This can be done by accessing the task interfaces described in Figure 7.3, i.e.

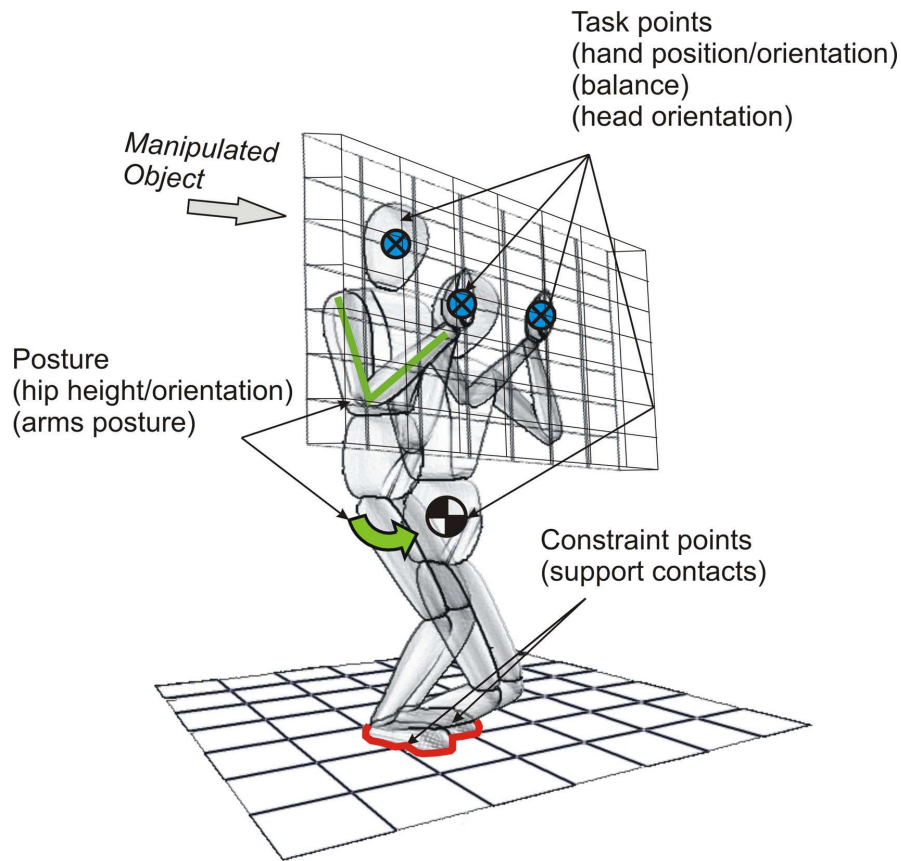


Figure 7.4: **Task decomposition:** We depict here a decomposition into low-level tasks of a whole-body multi-contact behavior. Each low-level task needs to be instantiated and controlled individually as part of the whole-body behavior.

```
handTask→maxVelocity(0.5);
handTask→maxAcceleration(3);
handTask→gain(400);
```

We also need to pass the desired task goal. For instance, if the goal is a teleoperated point we make the following calls

```
PrVector opGoal = worldModel→teleoperatedPoint();
handTask→goal(opGoal);
```

Here `PrVector` is an algebraic vectorial abstraction defined in our math library and `worldModel` is a pointer to a software module created to describe the robot's environment. In the above case, a haptic device is used to command desired hand positions with respect to a global

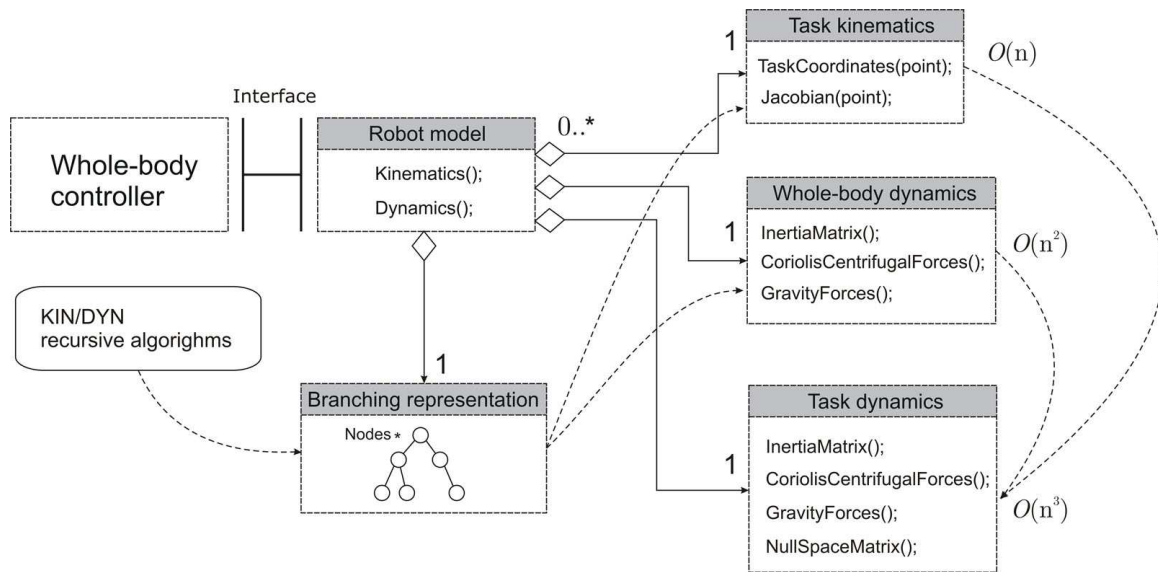


Figure 7.5: **Robot model:** UML class diagram describing robot kinematic and dynamic representations. The robot model provides access to kinematic and dynamic quantities both in joint and task spaces. It also contains a branching representation to compute these quantities recursively based on efficient kinematic and dynamic algorithms.

frame of reference. To finalize the instantiation of the task, we also need to indicate the desired priority level with respect to other operating tasks. This ordering will allow the controller to create prioritized control structures based on the algorithms we described in previous chapters. To indicate the priority we make the following call

```
handTask→priorityLevel(level);
```

We assign priorities based on the relative importance of each task with respect to the others. In general we divide primitives into different categories, each emphasizing its relative importance with respect to other categories. For instance, we consider the clustering of tasks shown in Figure 7.6 listed in decreasing order: (1) constraint handling primitives, (2) balance primitives, (3) operational primitives, and (4) posture primitives.

At every servo loop we update task representations and low-level controllers by making the following call

```
handTask→update();
```

which in turn executes the following calculations,

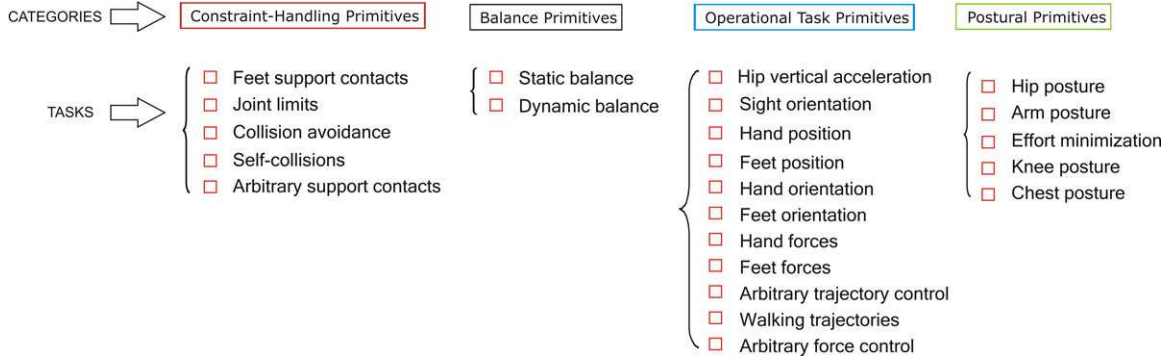


Figure 7.6: **Relative importance of task categories:** The above categories indicate the relative importance between tasks and are used to assign control priorities. The left most category has the highest priority since constraint-handling tasks ensure that the robot structure and the surrounding environment are not damage, while the right most category corresponds to the lowest priority level associated with the execution of postures.

```
void positionPrimitive::update() {
    calculateTaskState();
    calculateJacobian();
    calculateTaskDynamics();
    calculateControlRef();
};
```

In other words, it updates kinematic, dynamic, and control quantities of the task and calculates the control policy using the function `calculateControlRef()`. For the previous PD position control law with velocity saturation, the following control reference at the acceleration level is computed

$$a_{\text{hand}}^{\text{ref}} = \nu_a a_{\text{des}},$$

$$a_{\text{des}} = -k_v (\dot{x} - \nu_v v_{\text{des}}), \quad \nu_a = \min\left(1, \frac{a_{\text{max}}}{\|a_{\text{des}}\|}\right),$$

$$v_{\text{des}} = \frac{k_p}{k_v} \nabla(x - x_{\text{goal}}), \quad \nu_v = \min\left(1, \frac{v_{\text{max}}}{\|v_{\text{des}}\|}\right).$$

Here x is the right hand task coordinate computed with the robot model shown in Figure 7.5, and x_{goal} is the desired teleoperated goal obtained from the world model previously described.

7.1.3 Task Execution

While control primitives encapsulate task representation and control policies, the main task of the servo loop in the execution layer (see Figure 7.1) is to calculate control torques of all operating tasks and aggregate them together to create the desired whole-body behavior. To execute each task the following calls are made

```
handTask→jacobian(jacobian);
handTask→dynamicQuantities(inertia, ccForces, gravityForces);
handTask→priorityLevel(level);
handTask→controlRef(refAccel);
```

Here, kinematic and dynamic quantities are first obtained from the task primitive at hand, and the associated control policy is used to obtain the desired acceleration reference. For instance, for the previous hand position task where the priority level is equal to 3 according to the category ordering shown in Figure 7.6, the associated torque control vector as shown in the torque expression (5.6) is

$$\Gamma_{\text{tasks|p}(3)} = J_{\text{hand|p}(3)}^{*T} \left(\Lambda_{\text{hand|p}(3)}^* a_{\text{hand}}^{\text{ref}} + \mu_{\text{hand|p}(3)}^* + p_{\text{hand|p}(3)}^* \right), \quad (7.1)$$

where the subscript $\{\text{hand|p}(3)\}$ means that the hand task is controlled provided that balance and the acting constraints are first fulfilled and $a_{\text{hand}}^{\text{ref}}$ is the acceleration reference for right hand control based on the previous PD control law implementing velocity and acceleration saturation.

In general, when a set of low-level tasks are controlled as part of a whole-body behavior, the execution layer will produce the following torque output

$$\Gamma = \left(J_{\text{constraint}}^{*T} F_{\text{constraint}} \right) + \left(J_{\text{balance|p}(2)}^{*T} F_{\text{balance|p}(2)} \right) + \left(J_{\text{tasks|p}(3)}^{*T} F_{\text{tasks|p}(3)} \right) + \left(J_{\text{postures|p}(4)}^{*T} F_{\text{postures|p}(4)} \right), \quad (7.2)$$

where each task will be instantiated and used as an individual object as we did for the previous hand task.

7.2 Composition and Instantiation of Whole-Body Behaviors

We will develop here computational entities for the composition and creation of whole-body behaviors. When properly coordinated, these entities will serve as the main units of action

of a high level controller.

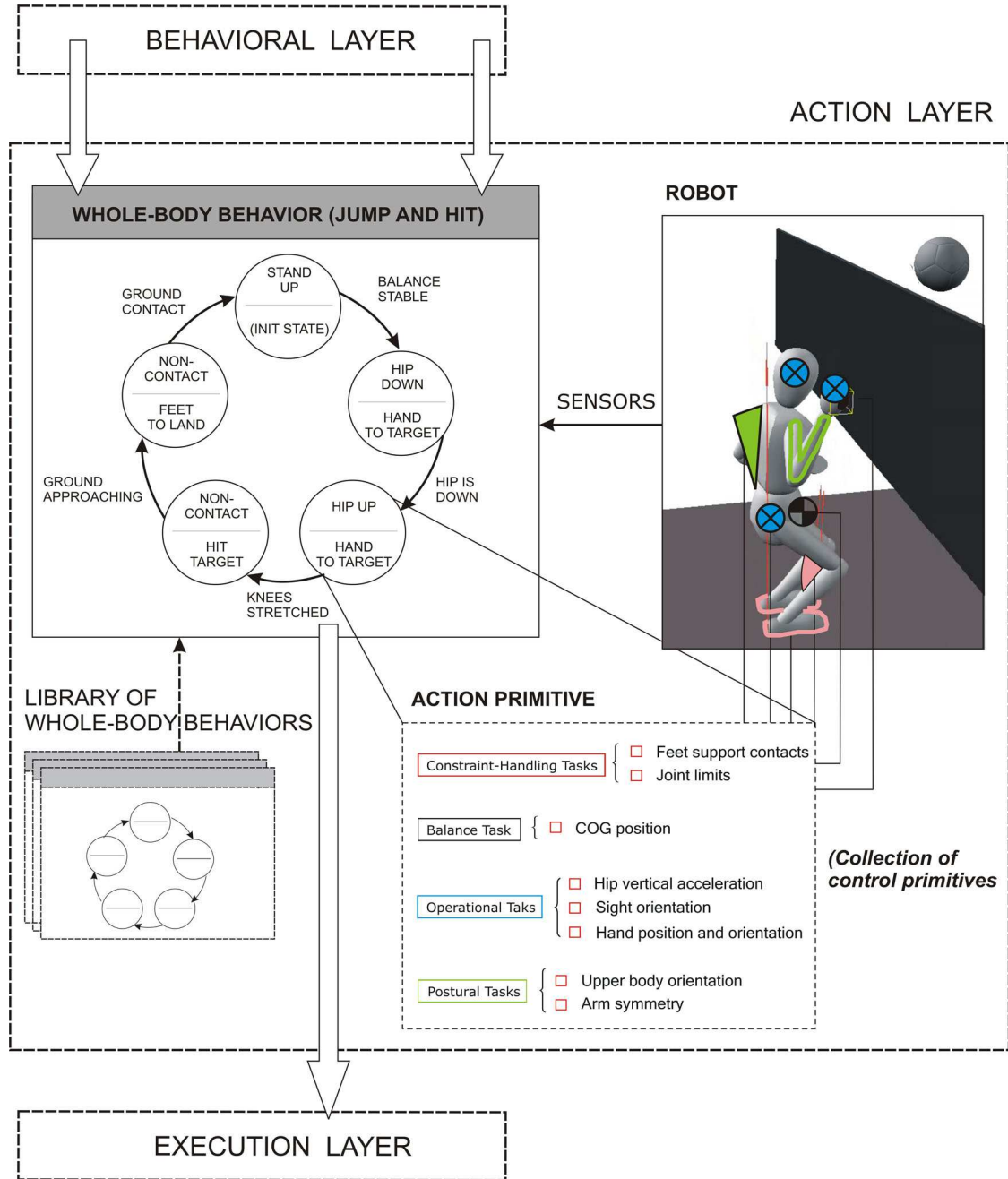


Figure 7.7: **Operation of the movement layer:** The centerpiece of this layer is a representation of whole-body behaviors as sequences of actions implementing different movement phases.

If the goal of the previous section was to abstract the representation of low-level tasks, the goal of this section is to abstract the representation of whole-body behaviors. This level of abstraction is aimed at providing meaningful units of action that encapsulate task decomposition and movement sequencing. Whole-body behaviors allow us to define, aggregate, and sequence collections of tasks into single units of action.

In Figure 7.1 we illustrate the operation of the action layer. This layer defines whole-body behavior representations. A whole-body behavior is a sequence of goal-oriented actions coordinated to achieve a global behavior. For instance, the volleyball jumping behavior depicted in the previous figure consists on five unique movement phases: (1) stand-up, (2) move the hip down, (3) accelerate the hip upwards, (4) hit the target, (5) prepare to land, and go back to standing up (1). Transitions between movements are predetermined and triggered by sensory events. Action primitives encapsulate task decomposition and coordination. For instance, a primitive used to accelerate the robot’s body upwards as in the previous example would involve simultaneously coordinating balance, hand position and orientation, head control, and posture control.

7.2.1 Action Primitives

An action primitive is an abstraction that encapsulates task decomposition and coordination. For instance, the primitive shown in the table below is used to create the previous

Action Primitive (Jumping Movement)

<i>Control Primitive</i>	<i>Priority Level</i>	<i>Control Parameters</i>
Obstacle Avoidance	1	$(d_{\text{safe}} = 0.1m, k_p = 800, v_{\text{max}} = 2m/s)$
Static balance	2	$(x_{\text{goal}} = C_{\text{feet}}, k_p = 1000)$
Hip height	3	$(x_{\text{goal}} = in, v_{\text{max}} = in, k_p = 400)$
Head orientation	3	$(\phi_{\text{goal}} = \hat{u}_{\text{ball}}, k_p = 100, v_{\text{max}} = 2\pi rad/s)$
Right hand position	3	$(x_{\text{goal}} = in, \dot{x}_{\text{goal}} = in, k_p = in)$
Right hand orientation	4	$(\phi_{\text{goal}} = in, \omega_{\text{goal}} = 2\pi rad/s, k_p = 100)$
Upper-body orientation	4	$(\phi_{\text{goal}} = \hat{u}_{\text{upright}}, \omega_{\text{goal}} = \pi rad/s, k_p = 100)$
R/L arm posture	4	$(q_{\text{arm}} = Q_{\text{human}}, k_p = 100)$

jumping behavior. Here, d_{safe} stands for a safety threshold to arbitrary obstacles, C_{feet} represents the center of the feet supporting polygon, \hat{u}_{ball} is the direction of sight towards the ball, \hat{u}_{upright} is an upright orientation vector, Q_{human} is a captured human pose, and

the symbol *in* means an input parameter provided at runtime by the sensory layer.

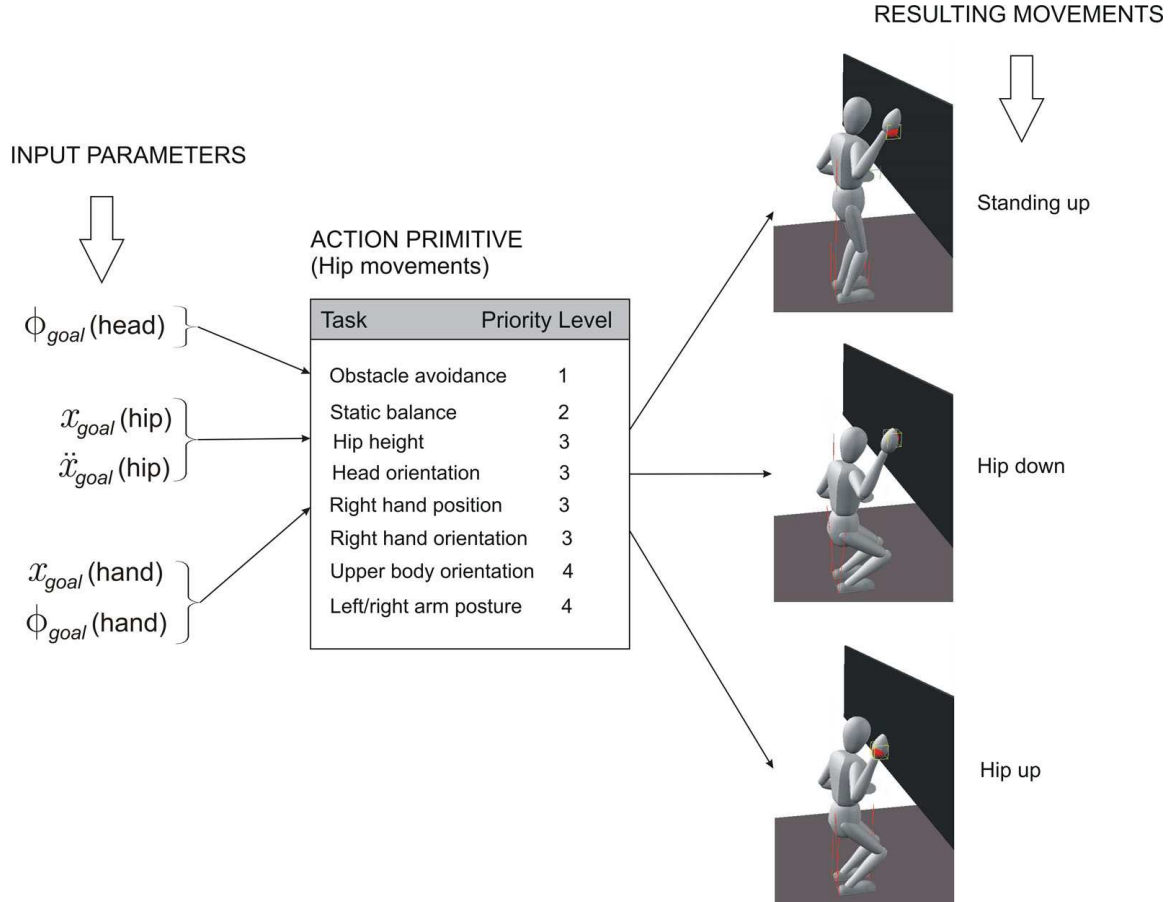


Figure 7.8: **Instantiation of movements:** By providing different input parameters we synthesize different type of movements at runtime.

In fact, action primitives serve as platforms to implement a variety of movements depending on the desired goals as shown in Figure 7.8.

7.2.2 Whole-Body Behaviors

We create whole-body behaviors by sequencing action primitives. With the proper sequencing and goals, the desired behavior emerges. For instance, let us consider the two movements shown in Figure 7.9 which are part of the jumping behavior shown in Figure 7.7. To accelerate the hip upwards we use an action primitive that involves the control of the hip's vertical position as part of the overall movement. When the knees reach full stretch, the next phase is triggered loading a new action to hit the ball in mid air. This

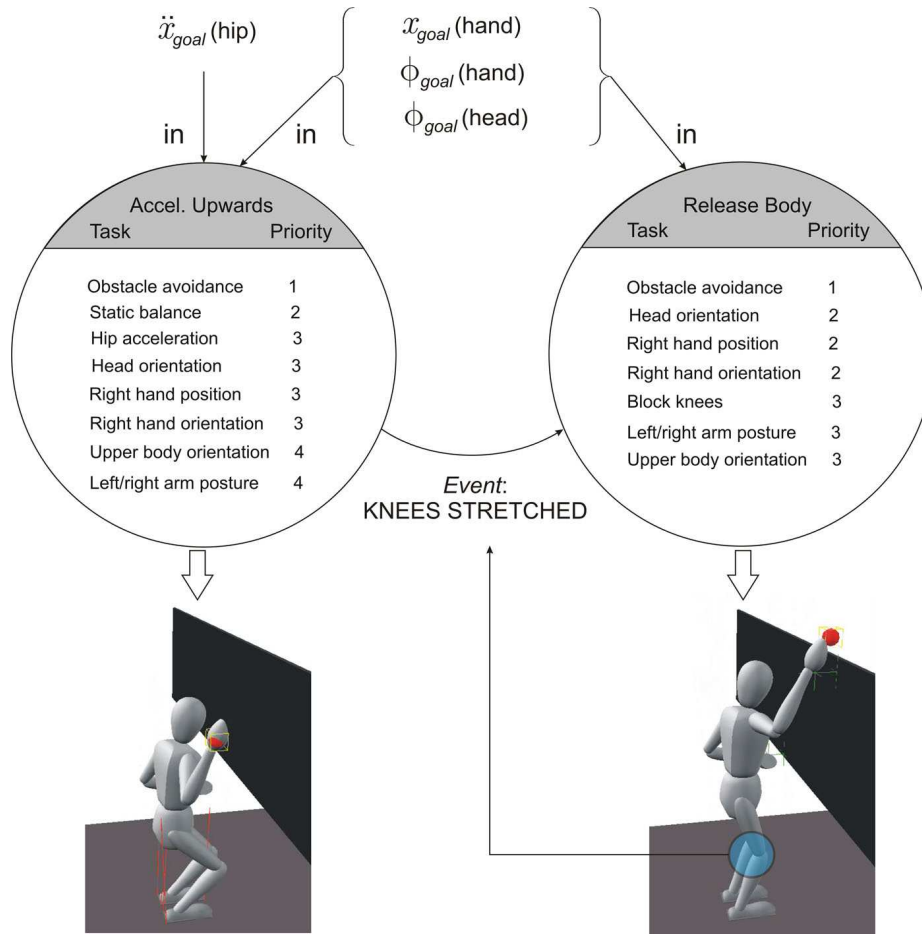


Figure 7.9: **Action sequencing:** This figure illustrates two actions used to create a jumping behavior. Initially an action primitive to accelerate the hip upwards is used. When the knees are fully stretched, a new action is loaded to control the body in mid air.

second action implements control of the robot’s right hand. The goals to accelerate the hip upwards and to hit the ball in mid air are provided at runtime by the sensory layer.

7.2.3 Behavior Feasibility

In previous chapters we discussed behavior feasibility and proposed metrics to measure it. For instance, when jumping in the previous example the task becomes infeasible if joint limits are reached while accelerating the body upwards. To modify the robot’s behavior in case of conflicting situations such as the previous one we create additional safety procedures. For instance, in Figure 7.10 we illustrate a more elaborated state machine where safety

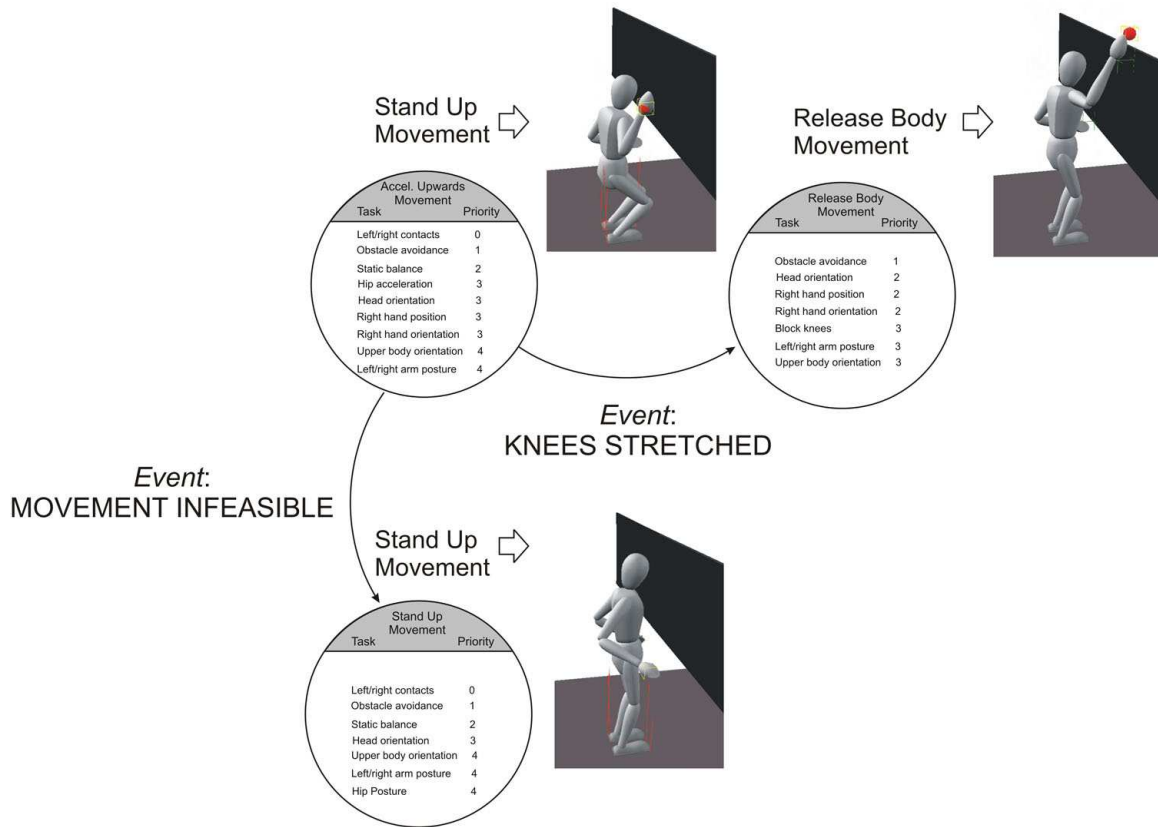


Figure 7.10: **Handling of infeasible tasks:** The two upper actions in the above figure are equivalent to the actions described in Figure 7.9. However, an additional state is added to handle conflicting scenarios where joint limits on the knees are reached while moving upwards.

actions are implemented to land safely in case of conflict.

Chapter 8

Concluding Remarks

Throughout this dissertation we have developed a whole-body control framework and supporting behavioral entities for the realtime synthesis of complex behaviors in humanoid systems. In this final chapter we will discuss some of the key points of our work and provide concluding remarks.

8.1 Summary of Results and Contributions

Looking back, we have addressed several key areas on control of humanoid systems. We extended the operational space formulation (Khatib 1987) to handle arbitrary task points under arbitrary supporting contacts and we proposed novel hybrid position/force control strategies for humanoid systems. We developed prioritized and non-prioritized multi-task control strategies that could control all aspects of motion. We developed techniques to respond in realtime to dynamic constraints and to measure task feasibility under dynamic constraints. Finally, we developed control and behavioral abstractions to support the creation of complex behaviors.

8.1.1 Whole-Body Control Methods

In whole-body control, we have focused on four different subjects: (1) The control of low-level tasks under supporting contacts, (2) the coordination and control of collections of tasks as part of whole-body behaviors, (3) the control of postural behavior, and (4) the control of movements in midair.

Task Control Under Supporting Contacts and Balance Constraints

- Based on the characterization of humanoids as underactuated systems with 6 passive DOFs attached to their base, we have developed novel task kinematic and dynamic representations under supporting constraints. In particular, studying the effect of supporting constraints we have developed representations of task velocities in terms of joint velocities alone and we have derived associated constrained Jacobians. Based on these representations we have developed operational space controllers for arbitrary task points that provide linear control of task forces and accelerations under supporting contacts. Our contributions on this subject are on characterizing constrained kinematic and dynamic representations under arbitrary supporting contacts, on developing novel operational space controllers to accomplish arbitrary task goals, on characterizing the residual movement redundancy associated with the task at hand, and on outlining strategies for controlling internal forces between supporting limbs.
- Using the above control representations, we have developed balance controllers to achieve static and dynamic balance based on direct manipulation on COG accelerations. Our contributions here are on developing controllers for linear control of COG accelerations and on developing controllers for linear control of ZMP positions.

Prioritized and Non-Prioritized Multi-Task Control

- We have develop torque controllers to simultaneously accomplish multiple task goals as part of whole-body behaviors. We have develop non prioritized structures where multiple tasks are represented as single macro tasks and controlled using operational space control methods. To decouple high priority tasks from lower priority tasks and to resolve conflicting scenarios between tasks we have also developed complementary prioritized controllers where lower priority tasks are projected and controlled in the null space of higher priority tasks. Our main contributions here are on developing controllers that provide linear control of task forces and accelerations while operating in the residual redundancy of higher priority tasks, on developing recursive expressions of prioritized Jacobians and null space representations, and on developing techniques to measure task feasibility under prioritized hierarchies.
- We have analyzed the decomposition of whole-body behaviors into low-level tasks for a variety of control examples and implemented multi-task controllers to accomplish desired behavioral responses. Some of the examples shown included hybrid

force/position control to track the contour of an object while simultaneously executing complex postural behaviors as well as a variety of static and dynamic walking behaviors.

Posture Control

- To control the robot's postural behavior, we have characterized the motion space that complies simultaneously with supporting constraints and prioritized tasks and use it to optimize performance criteria in posture space or to control position and orientation of postural DOFs without interrupting the global task. Our contributions here are on developing controllers that identify the controllable direction in posture space and use them to optimize gradient descent of postural criteria.
- Based on these methods we have presented novel techniques to imitate captured human poses without interrupting the global task. In particular, we have proposed postural criteria to minimize the joint-space distance to captured poses. We have also proposed a novel technique to minimize actuation effort in posture space. Associated criterions are based on previous observations of human behavior and consists on minimizing the weighted norm of gravity torques. By descending the gradient we have demonstrated that humanoid robots assume human-like postures.
- Based on dynamic compensation of posture behavior, we have demonstrated that we can choose arbitrary posture gains without losing postural performance. Low gains imply high posture compliance while high gains imply stiff postures.

Whole-Body Control of Movements in Midair

- We have developed kinematic and dynamic representations of humanoids during movements in mid air based on the conservation of the system's momentum. We have developed non-prioritized and prioritized multi-task controllers to handle whole-body movements in mid air akin to the controllers developed for ground based movements. Our contribution here is on providing a whole-body control framework for the realtime synthesis of movements in midair.
- When the robot is not in contact with the ground, the COG's trajectory cannot be modified. We have formulated feasibility measurements to monitor task feasibility during movements in free space.

8.1.2 Reactive Handling of Dynamic Constraints

We have developed a variety of techniques to respond quickly to dynamic changes in the environment and to deal with internal constraints imposed by the robot's own motion.

- We have developed novel control methods to handle dynamic constraints without interrupting the global task. In contrast with other approaches where constraints are handled as secondary tasks, we have addressed the handling of constraints as priority tasks while other operational tasks are projected in the null space of the acting constraints to prevent potential violations. Our contribution here is on proposing prioritization of constraints and on developing controllers that can accomplish task goals while operating in the constraint consistent residual space of motion. We have also proposed feasibility measurement quantities that can monitor task feasibility under the acting constraints and can be used to change robot behavior at runtime as part of a high level decision layer.
- We have also developed reactive techniques to respond to a variety of constraints in realtime including joint limits, obstacle avoidance, and self-collision avoidance.

8.1.3 Synthesis of Whole-Body Behaviors

To support the synthesis of emergent behaviors, we have developed control and behavioral abstractions that encapsulate task representations and action mechanisms.

- We have developed novel control abstractions called control primitives that support the instantiation of low-level tasks. These primitives encapsulate kinematic and dynamic representations as well as desired control policies. Instead of being based on predefined trajectories, these primitives represent a direct mapping between arbitrary task goals and actuation commands.
- We have developed novel behavioral abstractions called whole-body behaviors. These abstractions encapsulate task decomposition and action sequencing to achieve complex whole-body behaviors.

8.1.4 Implementation Details

The whole-body control framework developed in this thesis has been designed to run in realtime as part of the central control process of humanoid systems. We will discuss here some implementation details.

- We have developed a software architecture that implements the proposed control framework as well as the behavioral abstractions earlier described. This software implements a unified whole-body controller that controls simultaneously multiple operational tasks while automatically adapting to new contact conditions and dynamic constraints. It also provides a library of whole-body behaviors to be orchestrated by a high level control layer. Whole-body behaviors are implemented as state machines where the states are sets of low-level tasks simultaneously operated to accomplish individual movement actions.
- Computing kinematic and dynamic quantities in our control framework is computationally expensive for a realtime controller, with operations ranging from $O(n)$ to $O(n^3)$ of algorithmic complexity. To handle these computations in realtime, we implement two separate update loops. While feedback controllers are updated at servo rates, around $1KHz$, kinematic and dynamic quantities that depend only on joint positions are updated at slow rates, normally around $20Hz$ to $50Hz$. This dual update strategy is currently under test.
- To increase the computational speed associated with kinematic and dynamic quantities, we use the set of efficient kinematic and dynamic algorithms described in (Chang and Khatib 2000). Using these algorithms we compute kinematic quantities with $O(n)$ complexity, and whole-body dynamic quantities with $O(n^2)$ complexity.
- We have recently begun implementing the proposed whole-body control framework into Honda's Asimo robot. Because many humanoid robots are controlled through joint positions and not through torques, Khatib et al. recently developed a torque to position transformer (see Khatib, Thaulaud, and Park 2006). Here, joint positions are calculated through the transformation of torques based on the characterization of servo motor transfer functions. We have recently succeeded to control whole-body torque commands in the real humanoid and we are now in a position to start implementing the algorithms developed throughout this thesis.
- We have successfully tested all concepts of this dissertation into simulated humanoid systems. To simulate multibody dynamics and contact interactions we have used the Arachi simulation engine, an environment developed by former students of our lab K.C. Chang and D. Ruspini (Chang and Khatib 2000; Ruspini and Khatib 2000).

8.2 Discussion

Although our whole-body control framework is fairly complete and successfully tested in simulation, several issues need to be addressed.

In Chapter 2 we developed control structures to control internal forces between supporting links. However, no methods for controlling the actual forces were discussed. We are currently developing novel control methods to handle internal forces and moments.

We have described various techniques to measure task feasibility under the acting constraints. The condition numbers of constrained task Jacobians or inertias were proposed to characterize task feasibility. Normal values of these quantities need to be empirically characterized.

In Chapter 4 we developed control techniques for whole-body effort minimization. However, some of the motions did not look natural. To solve this problem, it would be more effective to minimize effort using only a few DOFs. For instance, effort could be minimized using knee, ankle, and hip joints only.

An open issue concerns the control of arm postures. A candidate posture would involve a weighted combination of effort minimization and self-collision avoidance but has not been implemented yet.

In Chapter 5 we described reactive techniques to avoid obstacles based on measuring close points between the robot and nearby obstacles. However, multiple points in different links could become close to objects leading to possible oscillations on the movement. The transition between one or more avoidance behaviors should be smoothen with additional techniques.

8.3 Summary of Publications

Although much of the theory and examples described throughout this dissertation is not yet published, here is a list of some publications generated during this thesis.

In (Sentis and Khatib 2006) we published an early version of whole-body control based on the concepts described in Chapters 2, 3, and 5. In (Sentis and Khatib 2005b) we published recursive control representations for multi-task control corresponding to some of the concepts described in Chapter 3. In (Sentis and Khatib 2005a) we published underactuated representations of humanoids and whole-body controllers for movements in mid air, corresponding to some of the concepts described in Chapter 6. Finally, in (Khatib, Sentis, Park, and Warren 2004) we published whole-body control foundations and posture controllers corresponding to some of the concepts described in Chapters 2 and 4.

8.4 Future Work

Implementation into Real Humanoids

We are currently implementing whole-body control into a full humanoid robot. At this stage we have succeeded in implementing torque to position transformations and whole-body torque controllers. We recently started to implement the concepts described throughout this thesis.

Towards Emergent Behaviors

The methods and software entities we have developed throughout this dissertation are aimed at supporting the synthesis of emergent behaviors in dynamic environments. Obviously this is a very ambitious task. To move on this direction we plan to further extend the capabilities of the proposed behavioral structures presented in Chapter 7. First, we plan to integrate perception and decision processes. For instance, vision systems will provide information about the environment. Contact and surface pressure sensors will provide information on contact interactions and support the localization of objects. Gyroscopes and accelerometers will estimate the orientation of the robot in space and the acceleration of the robot's COG, especially to implement effective running and jumping behaviors. On the other hand, decision processes will be implemented to achieve global behaviors in response to the environment.

Beyond creating intelligent modules that can sense and respond to the environment we will seek to develop architectures that can operate to achieve greater goals. This research connects with the work developed by Brooks on distributed behaviors (Brooks et al. 2004). We will pursue to scale up our control framework to a decentralized behavioral framework for the synthesis of emergent behaviors.

Learning Skills

To create new behaviors we have spent hours if not days to design movement sequences and tune up parameters to achieve the desired behaviors. Machine learning techniques will be needed to acquire complex skills (see reinforcement learning on quadrupeds (Stone 2000) and learning by imitation on full humanoids (Schaal et al. 2003)). Learning techniques applied to our control framework will benefit from the modularity of our structures and the automatic compliance with internal constraints.

Support for Motion Planning

The methods we have developed to respond reactively to dynamic events are meant to be implemented as stand-alone behaviors but could also be implemented as part of path relaxation techniques to support motion planners. For instance, given a candidate path, we can create an elastic strip and deform it using obstacle and self-collision avoidance potentials as well as joint limit blocking potentials. When a candidate path is obtained, the candidate trajectories can be executed in posture space, thus automatically complying with balance and supporting contacts.

Related Applications

It is particularly important the application of robotic algorithms to the study of biological systems, in particular to the simulation and modeling of musculoskeletal function (Delp and Loan 2000). This area of research is currently very active having a substantial impact on the understanding of some neurodegenerative diseases, on the design of tendon related surgical procedures, and on the characterization of the human motor control system.

Another important area of research is computer animation. Many laboratories and studios developing 3D animation techniques are actively incorporating robotic methods to synthesize more realistic and responsive human like behaviors. Two big driving forces are demanding further research on this direction: the motion picture industry and the video game industry. However, other industries of interest concern education and visual media which also require realistic human-like simulations to better convey information content.

Another important application is the simulation of workspaces for ergonomic design, as well as the simulation of humans operations in factory environments.

Appendix A

Mathematical Proofs

Property A.0.1 (Null space cross product commutation). *The order of multiplication between null space terms is irrelevant, i.e.*

$$\forall i, j, \quad N_{i|\text{prec}(i)}^{*T} N_{j|\text{prec}(j)}^{*T} = N_{j|\text{prec}(j)}^{*T} N_{i|\text{prec}(i)}^{*T}. \quad (\text{A.1})$$

Proof.

1. Developing the LHS of the above product we obtain

$$\begin{aligned} N_{i|\text{prec}(i)}^{*T} N_{j|\text{prec}(j)}^{*T} &= I - J_{i|\text{prec}(i)}^{*T} \bar{J}_{i|\text{prec}(i)}^{*T} - J_{j|\text{prec}(j)}^{*T} \bar{J}_{j|\text{prec}(j)}^{*T} \\ &\quad + J_{i|\text{prec}(i)}^{*T} \bar{J}_{i|\text{prec}(i)}^{*T} J_{j|\text{prec}(j)}^{*T} \bar{J}_{j|\text{prec}(j)}^{*T}. \end{aligned} \quad (\text{A.2})$$

Using the expression of $\bar{J}_{i|\text{prec}(i)}^{*T}$ given in (3.36) and the expression of $J_{i|\text{prec}(i)}^*$ given in (3.14), we can further develop the last term of the above equation into

$$J_{i|\text{prec}(i)}^{*T} \Lambda_{i|\text{prec}(i)}^* J_i^* N_{\text{prec}(i)}^* \Phi^* J_{j|\text{prec}(j)}^{*T} \bar{J}_{j|\text{prec}(j)}^{*T}. \quad (\text{A.3})$$

Let us assume that the i -th task has higher priority than the j -th task. We can then write the following expression based on the null space decomposition of Equation (3.32)

$$N_{\text{prec}(i)}^{*T} = N_{\text{prec}(j)}^{*T} N_{j|\text{prec}(j)}^{*T} \prod_{l=j+1}^{i-1} N_{l|\text{prec}(l)}^{*T}. \quad (\text{A.4})$$

Using the above decomposition we can further develop Equation (A.3) into

$$J_{i|\text{prec}(i)}^{*T} \Lambda_{i|\text{prec}(i)}^* J_i^* \prod_{l=j+1}^{i-1} N_{l|\text{prec}(l)}^* N_{j|\text{prec}(j)}^* N_{\text{prec}(j)}^* \Phi^* J_{j|\text{prec}(j)}^{*T} \bar{J}_{j|\text{prec}(j)}^{*T}. \quad (\text{A.5})$$

Let us assume that the equality $N_{\text{prec}(j)}^* \Phi^* = \Phi^* N_{\text{prec}(j)}^{*T}$ holds (see Property A.0.3). Using the equality $\left(N_{i|\text{prec}(i)}^{*T}\right)^2 = N_{i|\text{prec}(i)}^{*T}$ given in Property A.0.2, the above equation becomes

$$J_{i|\text{prec}(i)}^{*T} \Lambda_{i|\text{prec}(i)}^* J_i^* \prod_{l=j+1}^{i-1} N_{l|\text{prec}(l)}^* N_{j|\text{prec}(j)}^* \Phi^* J_{j|\text{prec}(j)}^{*T} \bar{J}_{j|\text{prec}(j)}^{*T}, \quad (\text{A.6})$$

where we have used the expression $J_{j|\text{prec}(j)}^{*T} = N_{\text{prec}(j)}^{*T} J_j^{*T}$ (3.14). It is easy to demonstrate that

$$N_{j|\text{prec}(j)}^* \Phi^* J_{j|\text{prec}(j)}^{*T} = 0 \quad (\text{A.7})$$

by using the expressions of $N_{j|\text{prec}(j)}^*$ shown in (3.33), the expression of $\bar{J}_{j|\text{prec}(j)}^{*T}$ shown in (3.36), and the expression of $\Lambda_{j|\text{prec}(j)}^*$ shown in (3.21).

2. Therefore, Equation (A.2) becomes

$$N_{i|\text{prec}(i)}^{*T} N_{j|\text{prec}(j)}^{*T} = I - J_{i|\text{prec}(i)}^{*T} \bar{J}_{i|\text{prec}(i)}^{*T} - J_{j|\text{prec}(j)}^{*T} \bar{J}_{j|\text{prec}(j)}^{*T}. \quad (\text{A.8})$$

We can do a similar development for the RHS of Equality (A.1) obtaining the same result as above.

3. We can also demonstrate in a similar way the above equality holds when the i -th task has lower priority than the j -th task. Therefore we conclude that the Equality (A.1) holds. \square

Property A.0.2 (Idempotence of $N_{\text{prec}(k)}^{*T}$). *The following equality holds*

$$\left(N_{\text{prec}(k)}^{*T}\right)^2 = N_{\text{prec}(k)}^{*T}. \quad (\text{A.9})$$

Proof.

1. We first consider the following equality

$$\left(N_{i|\text{prec}(i)}^{*T}\right)^2 = N_{i|\text{prec}(i)}^{*T}, \quad (\text{A.10})$$

which can be demonstrated developing the following expression,

$$\left(N_{i|\text{prec}(i)}^{*T}\right)^2 = I - 2J_{i|\text{prec}(i)}^{*T}\overline{J}_{i|\text{prec}(i)}^{*T} + J_{i|\text{prec}(i)}^{*T}\overline{J}_{i|\text{prec}(i)}^{*T}J_{i|\text{prec}(i)}^{*T}\overline{J}_{i|\text{prec}(i)}^{*T}. \quad (\text{A.11})$$

When using the property of generalized inverses $J_{i|\text{prec}(i)}^{*T}\overline{J}_{i|\text{prec}(i)}^{*T}J_{i|\text{prec}(i)}^{*T} = J_{i|\text{prec}(i)}^{*T}$, Equation (A.11) becomes

$$I - J_{i|\text{prec}(i)}^{*T}\overline{J}_{i|\text{prec}(i)}^{*T} = N_{i|\text{prec}(i)}^{*T}. \quad (\text{A.12})$$

2. Using the recursive expression of $N_{\text{prec}(k)}^{*T}$ (3.32) and reorganizing terms using Property A.0.1 we can derive the following equality

$$\left(N_{\text{prec}(k)}^{*T}\right)^2 = \prod_{i=1}^{k-1} \left(N_{i|\text{prec}(i)}^{*T}\right)^2. \quad (\text{A.13})$$

Using Equation (A.10) the above expression becomes

$$\prod_{i=1}^{k-1} N_{i|\text{prec}(i)}^{*T} = N_{\text{prec}(k)}^{*T}, \quad (\text{A.14})$$

where once more we have rearranged terms using Property A.0.1 □

Property A.0.3 (Commutation of $N_{\text{prec}(k)}^{*T}$ with respect to Φ^*). *The following equality holds*

$$\Phi^* N_{\text{prec}(k)}^{*T} = N_{\text{prec}(k)}^* \Phi^*. \quad (\text{A.15})$$

Proof by induction.

1. We will first consider the null space expressions given in Equations (3.40) and (6.69),

$$N_{\text{prec}(k)}^* = I - \sum_{i=1}^{k-1} \overline{J}_{i|\text{prec}(i)}^* J_{i|\text{prec}(i)}^*, \quad (\text{A.16})$$

where $\overline{J}_{i|\text{prec}(k)}^*$ is defined in Corollary 3.2.4.

2. Let us demonstrate (A.15) for $k = 1$:

$$\Phi^* N_1^{*T} = \Phi^* \left(I - J_1^{*T} \overline{J}_1^{*T} \right) = \Phi^* \left(I - J_1^{*T} \Lambda_1^* J_1^* \Phi^* \right) = N_1^* \Phi^*. \quad (\text{A.17})$$

Here we have used the expression of \overline{J}_1^* given in (3.36).

3. Assuming (A.15) holds for an arbitrary k value, let us demonstrate it for $k + 1$:

$$\Phi^* N_{\text{prec}(k+1)}^{*T} = \Phi^* \prod_{i=1}^k N_{i|\text{prec}(i)}^{*T} \quad (\text{A.18})$$

Let us study the individual products,

$$\begin{aligned} \Phi^* N_{i|\text{prec}(i)}^{*T} &= \Phi^* \left(I - J_{i|\text{prec}(i)}^{*T} \overline{J}_{i|\text{prec}(i)}^{*T} \right) = \\ &= \Phi^* - \Phi^* J_{i|\text{prec}(i)}^{*T} \Lambda_{i|\text{prec}(i)}^* J_{i|\text{prec}(i)}^* \Phi^* = \\ &= \left(I - \overline{J}_{i|\text{prec}(i)}^* J_{i|\text{prec}(i)}^* \right) \Phi^* = N_{i|\text{prec}(i)}^* \Phi^*, \quad (\text{A.19}) \end{aligned}$$

where we have used the expression of $\overline{J}_{i|\text{prec}(i)}^*$ given in (3.36). Recursively applying this equality we can reverse the null-space terms of (A.18), yielding the equality

$$\prod_{i=1}^{k-1} N_{i|\text{prec}(i)}^* \Phi^* \quad (\text{A.20})$$

However, the null-space terms above are multiplied in reverse order. But as shown in (A.1) the product between null space terms can be reversed yielding the desired result, i.e.

$$N_{\text{prec}(k)}^* \Phi^* \tag{A.21}$$

□

Property A.0.4 (Compact expression of $N_{\text{prec}(k)}^*$). *The following compact form of the recursive null space matrix exists*

$$N_{\text{prec}(k)}^{*T} = I - \sum_{i=1}^{k-1} J_{i|\text{prec}(i)}^{*T} \overline{J}_{i|\text{prec}(i)}^{*T}. \quad (\text{A.22})$$

Proof.

1. Property 1: $\forall i$, $\overline{J}_{i|\text{prec}(i)}^{*T} N_{i|\text{prec}(i)}^{*T} = 0$. This equality can be demonstrated by visual inspection when using the expression $N_{i|\text{prec}(i)}^{*T} = \left(I - J_{i|\text{prec}(i)}^{*T} \overline{J}_{i|\text{prec}(i)}^{*T} \right)$ shown in (3.33), and applying the rules of generalized inverses.
2. Property 2: $\forall i$, $\overline{J}_{i|\text{prec}(i)}^{*T} N_{\text{prec}(i)}^{*T} = \overline{J}_{i|\text{prec}(i)}^{*T}$, i.e. the null-space term vanishes. This equality can be demonstrated as follows. First we use the expression of $\overline{J}_{i|\text{prec}(i)}^*$ given in (3.36) yielding

$$\overline{J}_{i|\text{prec}(i)}^{*T} N_{\text{prec}(i)}^{*T} = \Lambda_{i|\text{prec}(i)}^* J_{i|\text{prec}(i)}^* \Phi^* N_{\text{prec}(i)}^{*T}. \quad (\text{A.23})$$

Now, using the property $\Phi^* N_{\text{prec}(i)}^{*T} = N_{\text{prec}(i)}^* \Phi^*$ shown in (A.15), the expression of $J_{i|\text{prec}(i)}^*$ shown in (3.33), and the property $(N_{\text{prec}(i)}^*)^2 = N_{\text{prec}(i)}^*$ shown in (A.9), the last term of the above equation vanishes.

3. Property 3: $\forall i, j$, with $i < j$ (i.e. i has higher priority than j) $\overline{J}_{i|\text{prec}(i)}^* J_{j|\text{prec}(j)}^{*T} = 0$. This can be seen by writing the equality

$$J_{j|\text{prec}(j)}^{*T} = N_{\text{prec}(i)}^{*T} \prod_{l=i}^{j-1} N_{l|\text{prec}(l)}^{*T} J_j^{*T}. \quad (\text{A.24})$$

Applying Property 2 the term $N_{\text{prec}(i)}^{*T}$ vanishes, and applying Property 1 the term $\prod_{l=i}^{j-1} N_{l|\text{prec}(l)}^{*T}$ (which contains $N_{i|\text{prec}(i)}^{*T}$) cancels out.

4. Next, we will use induction to proof (A.22). We first demonstrate it for $k = 3$. First we expand the null-space expression into

$$\begin{aligned} N_{\text{prec}(3)}^{*T} &= N_1^{*T} N_{\text{prec}(2)}^{*T} = \left(I - J_1^{*T} \overline{J}_1^{*T} \right) \left(I - J_{2|1}^{*T} \overline{J}_{2|1}^{*T} \right) = \\ &= I - J_1^{*T} \overline{J}_1^{*T} - J_{2|1}^{*T} \overline{J}_{2|1}^{*T} + J_1^{*T} \overline{J}_1^{*T} J_{2|1}^{*T} \overline{J}_{2|1}^{*T}. \end{aligned} \quad (\text{A.25})$$

However, the last term is equal to zero because $\overline{J}_1^{*T} J_{2|1}^{*T} = 0$. Here we have used the expression $J_{2|1}^{*T} = N_1^{*T} J_2^{*T}$ and Property 1.

5. In general, for any k the positive products (like the last term of the above equation) will always contain products like the ones described in Property 3, which are always 0. □
-

Bibliography

- Albu-Schaffer, A. and G. Hirzinger (2002, May). Cartesian impedance control techniques for torque controlled light-weight robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington D.C., USA, pp. 667–663.
- Anderson, F. and M. Pandy (2001). Static and dynamic optimization solutions for gait are practically equivalent. *Journal of Biomechanics* 34, 153–161.
- Arai, H. and O. Khatib (1994, 81-84). Experiments with dynamic skills. In *Proceedings of The Japan-USA Symposium on Flexible Automation*, Kobe, Japan.
- Arai, H. and S. Tachi (1991, April). Dynamic control of a manipulator with passive joints in an operational coordinate space. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, USA.
- Arbib, M. (1981). *Perceptual Structures and Distributed Motor Control*. In: Brooks VB (ed) Handbook of physiology (Section 2: The nervous system, Vol. II Motor control, Part I). American Physiological Society, pp. 1449-1480.
- Arkin, R. (1998). *Behavior-Based Robotics*. Boston, MA: MIT Press.
- Baerlocher, P. (2001). *Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures*. Ph. D. thesis, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland.
- Baerlocher, P. and R. Boulic (1998, October). Task-priority formulation for the kinematic control of highly redundant articulated structures. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, Canada.
- Billard, A. (2000). Learning motor skills by imitation: a biologically inspired robotic model. *Cybernetics and Systems* 1(2), 155–193.
- Boulic, R. and R. Mas (1995). Inverse kinetics for center of mass position control and posture optimization. *Workshop in Computing Series* (234-249).

- Boulic, R. and R. Mas (1996). Hierarchical kinematic behaviors for complex articulated figures. *Advanced Interactive Animation*.
- Brock, O. and O. Khatib (2002). Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research* 21(12), 1031–1052.
- Brock, O., O. Khatib, and S. Viji (2002). Task-consistent obstacle avoidance and motion behavior for mobile manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, USA, pp. 388–393.
- Brooks, R. (1986, March). A robust layered control system for a mobile robot. *International Journal of Robotics and Automation* 2(1), 14–23.
- Brooks, R., L. Aryananda, A. Edsinger, P. Fitzpatrick, C. Kemp, U.-M. O'Reilly, E. Torres-Jara, P. Varshavskaya, and J. Weber (2004, March). Sensing and manipulating built-for-human environments. *International Journal of Humanoid Robotics* 1(1), 1–28.
- Buckley, C. (1986). *The application of continuum methods to path planning*. Ph. D. thesis, Stanford University, Stanford, USA.
- Chang, K. and O. Khatib (2000, April). Operational space dynamics: Efficient algorithms for modeling and control of branching mechanisms. In *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Chatila, R. (1981). *Systeme de Navigation pour un Robot Mobile Autonome: Modelisation et Processus Decisionnels*. Ph. D. thesis, Universite Paul Sabatier, Toulouse, France.
- Cole, A., J. Hauser, and S. Sastry (1989). Kinematics and control of multifingered hands with rolling contact. *IEEE Transaction on Automation and Control* 34(4), 398–404.
- Conti, F. and O. Khatib (2005, March). Spanning large workspaces using small haptic devices. In *IEEE WorldHaptics*, Pisa, Italy, pp. 183–188.
- Crowninshield, R. and R. Brand (1981). A physiologically based criterion of muscle force prediction in locomotion. *Journal of Biomechanics* 14, 793–801.
- Delp, S. and J. Loan (2000). A computational framework for simulating and analyzing human and animal movement. *IEEE Computational Science and Engineering* 2(5), 46–55.
- Dubowsky, S. and E. Papadopoulos (1993, October). The kinematics, dynamics, and control of free-flying and free-floating space robotic systems. *IEEE Transactions on Robotics and Automation* 9(5).

- Ehmann, S. and M. Lin (2000). Swift: Accelerated proximity queries between convex polyhedra by multilevel voronoi marching. Technical report, University of North Carolina at Chapel Hill.
- Espiau, B., F. Chaumette, and P. Rives (1992). A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation* 8(3), 313–326.
- Featherstone, R. (1987). *Robot Dynamics Algorithms*. Norwell, USA: Kluwer Academic Publishers.
- Featherstone, R., S. Thiebaut, and O. Khatib (1999, May). A general contact model for dynamically-decoupled force/motion control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, USA.
- Fichter, E. and E. McDowell (1980). A novel design for a robot arm. In *Advancements of Computer Technology*, pp. 250–256. ASME.
- Fujimoto, Y. and A. Kawamura (1996). Proposal of biped walking control based on robust hybrid position/force control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Volume 4, pp. 2724–2730.
- Full, R. (1993). Integration of individual leg dynamics with whole-body movement in arthropod locomotion. *Biological Neural Networks in Invertebrate Neuroethology and Robotics*, 3–20.
- Gilbert, E. G., D. W. Johnson, and S. S. Keerthi (1988). A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation* 4(2).
- Hanafusa, H., T. Yoshikawa, and Y. Nakamura (1981). Analysis and control of articulated robot with redundancy. In *Proceedings of IFAC Symposium on Robot Control*, Volume 4, pp. 1927–1932.
- Harada, K., H. Hirukawa, F. Kanehiro, K. Fujiwara, K. Kaneko, S. Kajita, and M. Nakamura (2004, September). Dynamical balance of a humanoid robot grasping an environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, pp. 1167–1173.
- Harada, K., S. Kajita, K. Kaneko, and H. Hirukawa (2004, November). An analytical method on real-time gait planning for a humanoid robot. In *Proceeding of the IEEE/RSJ International Conference on Humanoid Robots*, Los Angeles, USA, pp. 640–655.

- Hauser, K., T. Bretl, K. Harada, and J. Latombe (2006, July). Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Workshop on Algorithmic Foundations of Robotic (WAFR)*, New York, USA.
- Hirai, K., M. Hirose, Y. Haikawa, and T. Takenaka (1998). The development of Honda humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Volume 2, Leuven, Belgium, pp. 1321–1326.
- Hirose, S., T. Masui, and H. Kikuch (1985). Titan-iii: A quadruped walking vehicle. *Robotics Research*, 325–331.
- Hogan, N. (1987, March-April). Stable execution of contact tasks using impedance control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Raleigh, USA, pp. 1047–1054.
- Ishiguro, H. (2005, July). Android science. In *Proceedings on A CogSci 2005 Workshop on Toward Social Mechanisms of Android Science*, Stresa, Italy, pp. 25–26.
- Jain, A. and G. Rodriguez (1993, August). An analysis of the kinematics and dynamics of under-actuated manipulators. *IEEE Transactions on Robotics and Automation* 9(4).
- Kailath, T., A. H. Sayed, and B. Hassibi (2000). *Linear Estimation*. Prentice Hall.
- Kajita, S., F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa (2003a, September). Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 14–19.
- Kajita, S., F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa (2003b, October). Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, pp. 1644–1650.
- Kanehiro, F. and H. Hirukawa (2001, September). Online self-collision checking for humanoids. In *Proceedings of the 19th Annual Conference of the Robotics Society of Japan*, Tokyo, Japan.
- Kerr, J. and B. Roth (1986). Analysis of multifingered hands. *International Journal of Robotics Research* 4(4), 3–17.
- Khatib, O. (1980). *Commande Dynamique dans l'Espace Opérationnel des Robots Manipulateurs en Présence d'Obstacles*. Ph. D. thesis, l'École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France.

- Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research* 5(1), 90–8.
- Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research* 3(1), 43–53.
- Khatib, O. (2004). *Advanced Robotic Manipulation*. Stanford, USA: Stanford University. Class Notes.
- Khatib, O. and J. Maitre (1978, September). Dynamic control of manipulators operating in a complex environment. In *Proceedings of RoManSy'78, 3rd CISM-IFTToMM Symposium*, Udine, Italy, pp. 267–282.
- Khatib, O., L. Sentis, J. Park, and J. Warren (2004, March). Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics* 1(1), 29–43.
- Khatib, O., P. Thaulaud, and J. Park (2006). Torque-position transformer for task control of position controlled robots. Patent. Patent Number: 20060250101.
- Khatib, O., J. Warren, V. Desapio, and L. Sentis (2004, First edition). Human-like motion from physiologically-based potential energies,. In *Advances in Robot Kinematics*, pp. 149–163. Kluwer Academic Publishers.
- Krogh, B. (1984, August). A generalized potential field approach to obstacle avoidance control. In *Proceeding of the Internatioal Robotics Research Conference*, Betlehem, USA.
- Kuffner, J., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue (2003, October). Motion planning for humanoid robots. In *Proceedings of the International Symposium of Robotics Research*, Siena, Italy.
- Kuffner, J., K. Nishiwaki, S. Kagami, Y. Kuniyoshil, M. Inabal, and H. Inouel (2002, May). Self-collision detection and prevention for humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, USA, pp. 2265–2270.
- Kwon, S., W. Chung, Y. Youm, and M. Kim (1991, October). Self-collision avoidance for n-link redundant manipulators. In *Proceedings of the IEEE International Conference on System, Man and Cybernetics*, Charlottesville, USA, pp. 937–942.
- Latombe, J. (1991). *Robot Motion Planning*. Boston, USA: Kluwer Academic Publishers.

- Latombe, J. (1999). Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research* 18(11), 1119-1128.
- Laumond, J. and P. Jacobs (1994). A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation* 10(5), 577-593.
- Liegeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics* 7, 868-871.
- Lozano-Perez, T. (1981). Automatic planning of manipulator transfer movements. *Proceedings of the IEEE International Conference on System, Man and Cybernetics* 11(10), 681-698.
- Lozano-Perez, T. (1983). Spatial planning: a configuration space approach. *IEEE Transaction of Computers* 32(2), 108-120.
- Maciejewski, A. and C. Klein (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *International Journal of Robotics Research* 4(3), 109-117.
- Marchand, E. and G. Hager (1998, May). Dynamic sensor planning in visual servoing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Volume 3, Leuven, Belgium, pp. 1988-1993.
- Mataric, M. (2002). Sensory-motor primitives as a basis for learning by imitation: Linking perception to action and biology to robotics. *Imitation in Animals and Artifacts*, Kerstin Dautenhahn and Christopher Nehaniv, eds., MIT Press, 392-422.
- Merlet, J.-P. (1996, February). Redundant parallel manipulators. *Laboratory of Robotics and Automation* 8(1), 17-24.
- Minguez, J. and L. Montano (2004). Nearness diagram navigation (nd): Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation* 20(1), 45-59.
- Moravec, H. (1980). *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Ph. D. thesis, Stanford University, Stanford, USA.
- Mori, M. (2004, September-October). Uncanny valley. In *ROBOCON*, Volume 28, Seoul, Korea, pp. 49-51. (Originally reported in 1970 in "Energy" vol. 7, no. 4).
- Mussa-Ivaldi, F., N. Hogan, and E. Bizzi (1985, October). Neural, mechanical, and geometric factors subserving arm posture in humans. *The Journal of Neuroscience* 5(10), 2732-2743.

- Nakamura, Y., H. Hanafusa, and T. Yoshikawa (1987). Task-priority based control of robot manipulators. *International Journal of Robotics Research* 6(2), 3–15.
- Nelson, G. and R. Quinn (1998, May). Posture control of a cockroach-like robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, pp. 157–162.
- Nenchev, D., Y. Umetami, and K. Yoshida (1992, February). Analysis of a redundant free-flying spacecraft/manipulator system. *IEEE Transactions on Robotics and Automation* 8(1).
- Neo, E., K. Yokoi, S. Kajita, F. Kanehiro, and K. Tanie (2005). A switching command-based whole-body operation method for humanoid robots. *IEEE Transactions on Mechatronics* 10(5), 546–559.
- Nilsson, N. (1969). Mobile automation: An application of artificial intelligence techniques. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, Washington D.C. USA, pp. 509–520.
- Oda, M., K. Kibe, and F. Yamagata (1996, April). ETS-VII space robot in-orbit experiment satellite. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, USA.
- Palmer, L. and D. Orin (2006). 3d control of a high-speed quadruped trot. *Industrial Robot* 33(4), 298–302.
- Papadopoulos, E. and S. Dubowsky (1991, December). On the nature of control algorithms for free-floating space manipulators. *IEEE Transactions on Robotics and Automation* 7(6).
- Petrovskaya, A., J. Park, and O. Khatib (2007, April). Probabilistic estimation of whole body contacts for multi-contact robot control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Rome, Italy.
- Pieper, D. (1968). *The Kinematics of Manipulators Under Computer Control*. Ph. D. thesis, Stanford University, Stanford, USA.
- Pratt, G. (1995, August). Series elastic actuators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, USA.
- Pratt, J., P. Dilworth, and G. Pratt (1997, April). Virtual model control of a bipedal walking robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, USA.

- Quinlan, S. (1994). *Real-time Modification of Collision-free Paths*. Ph. D. thesis, Stanford University, Stanford, USA.
- Raibert, M. (1986). *Legged Robots that Balance*. MIT Press, Cambridge, Ma.
- Raibert, M., M. Chepponis, and H. Brown (1986, June). Running on four legs as though they were one. *IEEE Journal of Robotics and Automation* 2(2), 70–82.
- Roth, B., J. Rastegar, and V. Sheinmann (1973). On the design of computer controlled manipulators. In *First CISM IFToMM Symposium*, pp. 93–113.
- Ruspini, D. and O. Khatib (1999, October). Collision/contact models for dynamic simulation and haptic interaction. In *The 9th International Symposium of Robotics Research (ISRR'99)*, Snowbird, USA, pp. 185–195.
- Ruspini, D. and O. Khatib (2000). A framework for multi-contact multi-body dynamic simulation and haptic display. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Russakov, J., O. Khatib, and S. Rock (1995, May). Extended operational space formulation for serial-toparallel chain (branching) manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Nagoya, Japan, pp. 1056–1061.
- Russakov, J., S. Rock, and O. Khatib (1995, June-July). An operational space formulation for free-flying, multi-arm space robot. In *Preprints 4th International Symposium on Experimental Robotics*, Stanford, USA, pp. 278–283.
- Schaal, S., A. Ijspeert, and A. Billard (2003). Computation approaches to motor learning by imitation. *Philosophical Transactions - Royal Society of London Series B Biological Sciences* 358(1431), 537–548.
- Sentis, L. and O. Khatib (2005a, April). Control of free-floating humanoid robots through task prioritization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 1730–1735.
- Sentis, L. and O. Khatib (2005b, December). Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics* 2(4), 505–518.
- Sentis, L. and O. Khatib (2006, May). A whole-body control framework for humanoids operating in human environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, USA.
- Siciliano, B. and J. Slotine (1991, June). A general framework for managing multiple

- tasks in highly redundant robotic systems. In *Proceedings of the IEEE International Conference on Advanced Robotics*, Pisa, Italy, pp. 1211–1216.
- Stewart, D. (1965). A platform with six degrees of freedom (platform with six degrees of freedom for flight simulation in pilot training). *Proceedings of the Institution of Mechanical Engineers* 180(15), 371–378.
- Stone, P. (2000). *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. Ph. D. thesis, Carnegie Mellon University, Pittsburgh, USA.
- Sugihara, T. (2004). *Mobility Enhancement Control of Humanoid Robot based on Reaction Force Manipulation via Whole Body Motion*. Ph. D. thesis, University of Tokyo, Tokyo, Japan.
- Udupa, S. (1977). *Collision Detection and Avoidance in Computer Controlled Manipulators*. Ph. D. thesis, California Institute of Technology, Pasadena, USA.
- Umetami, Y. and K. Yoshida (1989, June). Resolved motion rate control of space manipulators with generalized Jacobian matrix. *IEEE Transactions on Robotics and Automation* 5(3).
- Vukobratovic, M. and B. Borovac (2004). Zero-moment point thirty five years of its life. *International Journal of Humanoid Robotics* 1(1), 157–173.
- Xu, Y. and T. Kanade (1992). *Space Robotics: Dynamics and Control*. Prentice Hall.
- Yamane, K. and Y. Nakamura (2003). Dynamics filter concept and implementation of online motion generator for human figures. *IEEE Transactions on Robotics and Automation* 19(3), 421–432.
- Yokoyama, K., H. Handa, T. Isozumi, Y. Fukase, K. Kaneko, F. Kanehiro, Y. Kawai, F. Tomita, and H. Hirukawa (2003, September). Cooperative works by a human and a humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 2985–2991.
- Yoshida, K. (1994). Practical coordination control between satellite attitude and manipulator reaction dynamics based on computed momentum concept. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Piscataway, USA, pp. 1578–1585.
- Yoshikawa, T. (1985). Manipulability of robotics mechanisms. *International Journal of Robotics Research* 4(2).