Semiotics of Motion: Toward a Robotics Programing Language

Nicolas Mansard

▶ **To cite this version:**

Nicolas Mansard. Semiotics of Motion: Toward a Robotics Programing Language. Automatic. Université Paul Sabatier - Toulouse III, 2013. tel-01061112

HAL Id: tel-01061112
https://theses.hal.science/tel-01061112

Submitted on 11 Sep 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Habilitation à Diriger des Recherches

présentée

devant l'Université de Toulouse

par

Nicolas MANSARD

Équipe d'accueil : GEPETTO (LAAS-CNRS, TOULOUSE)
École Doctorale : EDSYS

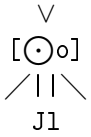
Titre de la dissertation :

Semiotics of Motion:
Toward a Robotics Programing Language

Vers une sémiotique de mouvement comme base d'un langage de programmation en robotique

Jury (à compléter)

Alessandro	DE LUCA	(Univ. Sapienza di Roma)	Rapporteurs
Philippe	FRAISSE	(Univ. Montpellier, LIRMM)	
Emanuel	TODOROV	(Univ. Washington)	
François	CHAUMETTE	(INRIA Bretagne)	Examinateurs
Jean-Baptiste	HIRIART-URRUTY	(Univ. Toulouse)	
Abderrahmane	KHEDDAR	(JRL-Japan)	
Jean-Paul	LAUMOND	(LAAS-CNRS)	



François CHAUMETTE: Directeur de recherche, INRIA Rennes Bretagne Atlantique
http://www.irisa.fr/lagadic/team/Francois.Chaumette-eng.html

Alessandro DE LUCA: Professor, Università di Roma "La Sapienza"
http://www.dis.uniroma1.it/~deluca/

Phillipe FRAISSE: Professeur des Universités, Université de Montpellier 2
www.lirmm.fr/~fraisse/

Jean-Baptiste HIRIART-URRUTY: Professeur des Universités, Institut des Mathématiques de Toulouse, Université Paul Sabatier
http://www.math.univ-toulouse.fr/~jbhu/

Abderrahmane KHEDDAR: Directeur de recherche, JRL-Japan, LIRMM

Jean-Paul LAUMOND: Directeur de recherche, LAAS-CNRS
http://homepages.laas.fr/jpl

Emanuel TODOROV: Associate Professor, University of Washington
http://homes.cs.washington.edu/~todorov

Contents

Chapter 1

General Introduction

The first reason to give to a robot a humanoid shape is because it is fun. In other words, the humanoid corresponds to the shape that people are expecting from a robot, coming from the mythology of perfect servant machines, from the first inventors (Da Vinci [DaVinci 19] or de Vaucanson) and from the early science-fiction literature. This expectation of the public gives a large visibility to research in humanoid robotics. It is used for example to highlight upstream research done by some large companies, by the mean of advertisement or public diffusion (see Fig. 1.1-(a)). The same effect is used to motivate undergraduate students by practical work around humanoid platforms, and is one of the reasons of the success of the Nao robot [Taïx 12].

The humanoid robot is generally considered to be an intuitive robot structure in the context of human-robot interaction. It is expected to provide a better understanding of the robot motion decision through the humanoid embodiment, in particular to non-expert users such as elderly: your grandmother is supposed to prefer a human-shaped robot companion to other functional shapes. Fig. 1.1-(b) gives some illustrations of human-friendly robots. If a complete *"cybernetics"* reproduction like Geminoid [Sakamoto 09] or HRP-4C [Kaneko 09] may not be essential for a good understanding of robot gestures, the reproduction of human schemes by humanoid robot is indisputably of some help. For example, Kobian uses its exaggerated face in coordination to the rest of its body to amplify some expression in a non-verbal communication [Zecca 09]. This aspect is of course debatable, since the interface-ergonomics progress seldom implies a biomimetic replication [Walters 08].

The humanoid kinematic shape is particularly interesting because it simplifies the programing of specific movements, off-line by motion capture or on-line by teleoperation. Fig. 1.1-(c) gives examples of human-based motion generation. The humanoid shape provides a very intuitive feedback to the distant teleoperator in [3]. With a very short training period (a few minutes), the operator is able to drive the robot, explore the room and handle objects, giving a true telepresence experience. The humanoid shape was selected for teleoperation in the International Space Station by the NASA [Diftler 11]. Teleoperation generally implies physical interactions between the human and a haptic device that displays the robot activity. The free

Asimo

Nao

Evolta

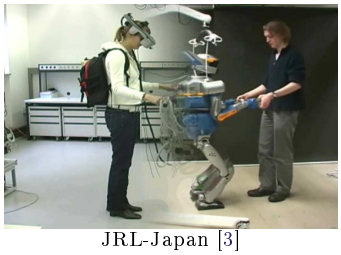
(a) Humanoid robotics is fun

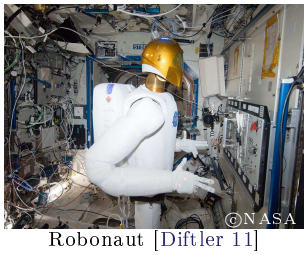
Kobian [Zecca 09]

Geminoid [Sakamoto 09]

HRP-4C [Kaneko 09]

(b) Your grandmother likes it

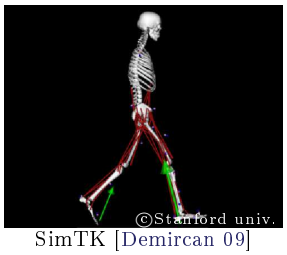

JRL-Japan [3]



Robonaut [Diftler 11]


Dance with HRP-2 [2]

(c) Anthropomorphism may simplify the motion programing



SimTK [Demircan 09]

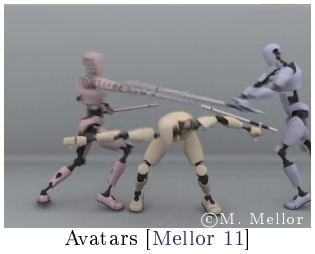
HAL [Nabeshima 12]

Ankle prosthesis [Au 09]

(d) There are close links with human motion

Alpha "*Bull*" Dog [Raibert 08]


Nextage [Ind 10]



Avatars [Mellor 11]

(e) Beyond anthropomorphism, generic problems are explored

Figure 1.1 – *Five good reasons to make a robot humanoid.*

human motion can also be recorded by motion capture and used as a basis to generate the motion of the robot [2], similarly to what has been done in computer animation for a long time [Ginsberg 83]. Once more, the humanoid structure is not essential to the adaptation of human motion but it truly helps to improve the intuition of the operator and to ease the transfer.

The human shape is also an inspiration by its structural organization. Humans reach far better movement capabilities than humanoid robots. The human body has more power, more energy, more accuracy and more resistance than actual robots: it can jump, fall, hit a nail with a hammer or play piano (not even speaking about cognitive capabilities). On the other hand, robotics has developed some computational motion techniques that are relevant tools to study the nature of the livings. The common anthropomorphic shape helps to bind both knowledge fields [Laumond 13]. The links can be extended to applications like ergonomics [Yang 04, Fourquet 07], prosthesis [Au 09] and exoskeleton for rehabilitation [Nabeshima 12, Strickland 12] or for augmentation [Zoss 06, Kazerooni 08].

Behind the anthropomorphic shape, the humanoid robot embeds several representative problems that can be found in other classes of robots and beyond. Locomotion of bipedal systems can similarly apply to quadruped [Pratt 06] and other legged systems [Raibert 84] or wheeled robot with an active control of the wheel position [Siegwart 02, Besseron 08]. Balance is also important for mobile robots on uneven terrains [Lamon 04] or large-arm mobile manipulators as soon as the ratio between the arm inertia and the basis surface becomes large [Stilman 10]. Multi-contact locomotion planning [Hauser 08] is important for outdoor robotics, both for legged [Hartikainen 92, Pongas 07] and mobile [Hait 02] robots. It has also many common aspects with dexterous manipulation [Han 98, Bouyarmane 10]. More generally, motion in contact is an important aspect of manipulator robot for tooling. The redundancy of the humanoid robot [Baerlocher 04] is a very good testbed for flexible cells of several fixed manipulator arms cooperating together on a same task [Caccavale 08, Basile 12]. The planning of multi-arm systems induces particular topological structures in the configuration space [Gharbi 09]. These structures are explored by algorithms first referred as *manipulation* planning [Siméon 04] and are typical of the humanoid trajectories [Dalibard 10, Berenson 11]. On the design aspect, the humanoid robot is a mobile robot that should be energy autonomous. The power limits lead to a reduction of the motor size [Enoch 12], which will finally benefit to human-friendly manipulators [Shin 10]. The humanoid platform also pushes developments in sensing [Davison 07] or human-robot interaction [MarinUrias 09]. Finally, the methods used to plan and control humanoid robot movements can be literally applied to generate realistic motion of virtual avatars [Baerlocher 04, de Lasa 10] and are one of the important classes of direct applications of our methods today.

Beside the real applicative interest of the humanoid structure, we rather see the humanoid robot as one of the most advanced robotics platform today, which gathers several issues that are typical of other robotic structures. In particular, we see the humanoid robot as a dynamic redundant mobile manipulator whose sensing capabilities and actuation power are limited. The methodologies that we will develop in this document answer to these problems with a generic approach and can therefore be applied to various other classes of robots. Typically, the work presented in the following may apply to humanoid robots and, behind, to quadruped or multi-legged robots, to redundant or multiple-cooperating manipulators, mobile manipulators, to virtual avatars with human or non-human shape and to biomechanical studies.

Chapter 2

Approaches and directions

This chapter presents the context of our work and introduces the contributions of this document.

2.1 Challenges

As explained in the general introduction, we see the humanoid robot as an experimental platform to develop and validate general concepts that may apply to different fields of robotics. We present some of these challenges that are linked to the work presented in the remaining of this document, with an arbitrary organization from bottom to top.

2.1.1 Mechatronics

The easiest solution to build a robot is to assemble series of electric direct-current motors that shape the kinematic tree [Kato 73]. The direct-current motors produces a high output velocity with comparatively limited torques. Gears with high ratio, such as harmonic drives, are used to reduce the speed and increase the output force, thus leading to smaller motors and smaller energy source. Examples of such humanoid robots are the Wabots [Ogura 06], Asimo [Hirai 98], the H5, H6, H7 [Nishiwaki 07] or HRP platforms [Kaneko 02, Kaneko 08]. However the reduction ratio similarly divides the external forces applied at the output of the gears and transmitted to the motor. Added to the friction forces inside the gear, this acts as a mechanical mask that prevents the motor from feeling the robot environment. Such setups are very efficient for position-based control. For all tasks requiring physical interaction (locomotion, balance, human safety), force control [Inoue 74] is preferable.

Indeed, the motor output torques are directly linked to the input current (for fixed voltage) and can then be reconstructed. However, the gear masks the joint dynamics from the motor dynamics and makes it difficult to reconstruct the joint output torques (gear output) from the motor output torques (gear input). It is possible to change the design of the mechanical structure to reduce the need of the high gear ratio. A solution is to reduce the gear ratio

by increasing the motor power. The motors can be set up higher in the kinematic tree, while the mechanical power is transmitted from this mechanical power source to the joints. Cable can be used, but for humanoid robots this does not fundamentally change the problem encountered with motors close to the joint: one motor has to be set for each joint in series with a reduction gear. The main advantage is to reduce the weight at the end of the kinematic chains [Townsend 99] but gears are still needed, with the cost of more complex mechanical integration. Examples are the legs prototype Sherpa [Olaru 09] and the future humanoid Romeo [Guizzo 10].

Alternatively, the mechanical power can be driven to the joint using a fluid, hydraulic like Petman [Buehler 05, Nelson 12] or air cylinders [Bobrow 98]. A single central pump is used and the fluid is driven to the cylinder attached to each joint. Like electrical motors, hydraulic actuation offers a rigid and high-bandwidth control. The additional power compared to electrical actuation [BostonDynamics 13] comes at the cost of a more complex integration [Habibi 00]. On the other hand, pneumatic actuation is elastic since the fluid can compress. The elastic factor prevents high-bandwidth control but at the same time is desirable for physical contact [Tassa 13]: any contact leads to a compression of the cylinder, which improves the contact stability and enables a measure of the contact forces from the internal position of the cylinder.

More generally, the same principle will be applied to any stiff actuation system: an elastic component is added in series with the actuation. Contacts will load the spring, smoothing the interaction and enabling measurement [Pratt 95]. The elasticity is desirable at the contact but is difficult to control, especially for fast free-space movements. An example of complete series-elastic actuation is the Coman robot [Tsagarakis 11].

A possibility is to reduce the elasticity while keeping the capacity of measuring the forces at the interface. A nearly-stiff and well calibrated material is added in series to the actuator and force gauge are added to measure the (supposedly infinitesimal) deformation. The kinematic structure is not totally rigid and care as to be taken when implementing the low-level controller to take the flexibility into account [AlbuSchäffer 07b] but the global system can be considered to be stiff. The desired impedance of the joint may then be tuned by the controller using the force back fed by the force sensor. The interaction bandwidth is of course limited by the frequency of the controller [AlbuSchaffer 07a], but modern electronics can reach a very satisfying rate [DeLuca 06]. In addition to this bandwidth limit, the drawback is the added cost coming from the force sensor. Obtaining an elastic contact is finally relatively easy, but the difficulty comes when trying to obtain a stiff behavior from the spring. Reducing the elasticity while keeping the measurement capabilities is interesting. However it cannot preserve the main properties of the elastic mechanisms: the possibility to store some energy at the impact and the inherent safety of the system. This major drawback is maybe in the initial contradiction: the system is built to be stiff with an important extra cost and finally the controller is used to obtain a compliant behavior.

Rather than reducing the flexibility from the conceptual design, another approach (sometimes considered as bio-inspired [Shin 10]) is to add a second motor on the same joint to tune the flexibility. The variable-stiffness concept [English 99] is copiously developed in several projects, with a very important pushing role of the Italian school [Tonietti 05, Visser 10, Flacco 12, Sardellitti 12, Jafari 13, Carloni 12].

The challenge to find the proper actuation system with enough power, integration and force-perception capabilities remains open and directly impacts on the following control issue.

2.1.2 Control

If we consider that the mechanical design is an answered problem, the control of the integrated system does not present a stronger fundamental challenge than the control of each separated actuator. Position-based control of the motor-plus-gears humanoid robots are straight-forward. If a force sensor is added to the kinematic chain, impedance-based control directly enables to take into account the force feedback [Hogan 85]. The challenge set by the integration of series-elastic or variable-stiffness robots seems to be similar in complexity (once more, if we consider that the actuator is integrated and controlled). The problematic brought by humanoid robots are rather on the size of the problem (number of variables to control, number of constraint to satisfy), on the consideration of the robot dynamics and the evolution of the trajectory describing the future of the robot.

The humanoid robot typically have around forty significant degrees of freedom to control during a movement (without considering finger, eyes or toe joints – 150 degrees of freedom to reach the human-body complexity [Nakamura 05]). On the other hand, it is subject to several constraints: under-actuation of the placement joint, feet on the ground, mobility constraint on the center of mass, on the joint limits, etc. Classical constraint resolution, such as inverse kinematics [Whitney 69, Liégeois 77, Baerlocher 04] directly applies to the humanoid robot structure. The size of the problem induces an important computation cost, which can becomes problematical for real-time control: the cost for an inverse-kinematics resolution is typically quadratic in both the number of parameters and the number of constraints [Golub 96]. Among the robot constraints, some depends on the dynamic variables, forces, joint torques or accelerations [Khatib 87, Collette 07]. Taking into account these additional variables and constraints again increases the size of the numerical problems.

In addition to the difficulties linked to the problem size, the problem solver also has to always output a realistic solution, since this solution is going to be used immediately to perform an action in the real world. In particular, when the constraints of the robot conflict with the objectives of the motion, the solution has to remain consistent with the robot capabilities and safe for its environment [Deo 92, Sugihara 11].

To keep a tractable computation cost, the robot evolution equations are often linearized around the instantaneous evolution (sometimes referred as *resolved motion rate* [Whitney 69] or *instantaneous Task Specification using Constraints* [Decré 09]). The instantaneous linearization cannot encompass the balance of the biped robot [Wieber 08]. For low-speed motion, the balance is reduced to a constraint keeping the center-of-mass inside the support polygon [Sugihara 02] but this approximation does not stand as soon as the robot acceleration cannot be neglected. Walking for example is achieved by injecting the robot main dynamics (*i.e.* the center-of-mass acceleration inside its horizontal plane) along an optimized trajectory [Kajita 03]. It is then a simple technological matter to be able to recompute this optimal trajectory and to use it as optimal control [Herdt 10].

The general idea can be applied in various contexts: we extract the main dynamic component of a motion and optimize it into a simplified context before injecting it into the linearized whole-body solver. Two problems then arise. First, extracting the main dynamics for each specific context might be difficult and lack of genericity. Second, it might be difficult to synchronize the decision taken by each simplified dynamics with the global linearized system. For example, in the walking context, it is difficult to decide what the next footsteps that satisfy a manipulation task [Dune 10, Kanoun 11b].

2.1.3 Trajectories

Alternatively, the future of the whole robot body can be considered inside the motion problem. Similarly as upper, the objective is written as a cost function depending on the states and controls of the robot, but this time over a short preview time interval. The cost function is minimized under constraints on the system dynamics and bounds. Mathematically, optimal control seems to be the most generic solution to write the problem of selecting a trajectory satisfying the motion objectives. This field is subject to extensive researches [Chevallereau 01, Arisumi 08, Ratliff 09, Schultz 10, Kalakrishnan 11, Pham 12, El Khoury 13, Lengagne 13].

If the trajectory computation is fast enough, it is possible (like for the center-of-mass walking pattern [Herdt 10]) to use it as a control, by recomputing the whole trajectory at each robot control cycle before applying only the corresponding first control. This is referred as model-predictive control [Diehl 09, Wieber 13]. Computationally speaking, model-predictive control is not as difficult as trajectory optimization, since at each control cycle, the problem only changes slightly and most of the computations of the previous control cycle can be reused.

However, this solution is subject to practical problems that for now prevent its application. First, the computation cost is yet prohibitive. For a humanoid robot like HRP-2, several minutes of computation are needed for seconds of trajectory optimization. Generally, the trajectory is sampled along an integration grid. The number of variables due to the dimension of the robot configuration space is multiplied by the number of samples along the grid (typically, hundreds of samples are chosen), which makes the problem difficult to solve in a short time. On the opposite, keeping the dimension of the configuration space is possible if preserving the functional nature of the trajectory [Claeys 12] but such solutions does not seems tractable for our typical system dimension. Added to this cost problem, the temporal dimension adds more degrees of freedom to the system. Instead of having to chose only among the *spacial* redundancy, the solver now has also to choose among all the possible trajectories [Pham 12]. This makes the numerical problems even more difficult. In particular, ensuring always a good behavior of the solution is very challenging.

A recent work [Tassa 12] gives a totally new view on this complexity problem, by managing to achieve computation times close to the real-time requirement while keeping a very relevant robot behavior during long time ranges. The methods is yet limited to virtual avatars, but with performances that may enable an application on real robots in a close future.

Finally, the trajectory optimization problem generally does not have any specific properties that can be exploited by the solver (exception can be found: the linearized inverse pendulum is linear [Herdt 10], whole-body trajectory optimization can be rewritten as a linear affine-transformation problem [Pham 12] or as a polynomial problem [Lengagne 13]). Non-linear solvers are then used, that require a valid initial guess. Finding one feasible trajectory is often as tricky as searching for the locally optimal one. Sample-based planning techniques are then used to find the initial guess [Barraquand 92, Laumond 98]. The humanoid robot is subject to constraints (contact stability, balance) that are respected on submanifolds of the configuration space. The planner is searching to sample this zero-measure submanifold [Berenson 11]. Since no explicit parameterization of these submanifolds exists, the whole configuration space is generally sampled and then projected onto the given submanifold [Kaiser 12]. Additionally, the submanifolds create a folded structure of multiple leaves in the configuration space, while the robot cannot directly move from one leaf to the other [Siméon 04].

2.1.4 The gap of planning

On the first hand, there are some complex methodologies to compute at high cost a trajectory of the robot in the configuration space. On the other hand, there are mechatronics and control development that enables the robust execution of dynamic movements. These two classes of methods seem complementary. However, there are some difficulties to go from the first to the second. On manipulator robots, a small disturbance on the trajectory in the configuration space leads to a small disturbance on the final position. The drift is possible to correct by servoing the robot configuration or a function of it [Samson 91]. The folded structure of the humanoid configuration space changes this stability property: an even-slight disturbance of the trajectory may lead to a change of leaf that makes the trajectory infeasible.

The problem can be seen from the sensory-feedback point of view. Depending on the modalities, several different sensors are relevant to observe and correct the robot motion. Consider the example of the robot walking through a door. The position of the landing foot might be important and can be servoed by vision. At the landing instant, the contact interface is more relevant and is observed by force sensors. During the flight, the position of the foot is less important but induces dynamical effects on the robot body that can be perceived by accelerometers. The swing of the shoulders is not directly observable by lack of sensors, but can be reconstructed from partial models of the environment to prevent a collision with the door. The robot thus has to combine multi-modal models of its environment perception, with different precision scales. Several sensor loops have to run simultaneously and sequentially. This means that a lot of information has to be added to the classical configuration trajectory output by upstream motion planners. The questions are even more complex when considering a model-predictive controller as the basic controller, since predictive sensor models have to be added [Allibert 10].

It seems that no effective purely-automatics solution has emerged yet to bind planning and control in humanoid robotics, while many works are trying to bridge the gap. Explicit decoration is added to the most evident part of the trajectories, enforcing for example a visual servoing of the robot hand to grasp an object [Morales 06, Vahrenkamp 09], or correcting the trajectory during the walk based on central vision-based localization [Moulard 12d]. In [Baudouin 11], the footstep planner is used during the robot control cycles to modify the trajectory based on sensor localization. Control from symbolic plans with geometrical reasoning is explored in [Gravot 06, Rusu 09].

In summary, the humanoid robot is a complex structure due to the number of various modalities of action that it is able to consider. Each modality can be handled separately by planning and control algorithms (one for manipulation, one for navigation, one for locomotion on rough terrain, etc). Searching for integrated algorithms able to handle several modalities with a same model is more complex.

2.1.5 Applications

Robotics, the robot companion: In the quest for robot autonomy, research and development in robotics are dominated by the stimulating competition between computer science and control theory, between abstract symbol manipulation and numerical signal processing. The humanoid robotics pushes one step further this quest, by emphasizing the importance of the numerical aspects. By the ambivalence of the mathematical *folded* structure of the contacts and the links between the locomotion and the manipulation modalities, the humanoid robot

proposes a renewal of the classical robotics challenges. The most representative application in this direction is the robot home companion [Dautenhahn 05], *friend, assistant or butler*. It is for example the avowed target of the future French humanoid Romeo.

Similarly, companion worker can be imagined [3]. The humanoid robot is then representative of the universal worker robot, able to accomplish a wide range of automatic tasks. Humanoid or not, those robots may composed the future flexible manufacturing cells [Basile 12], whose production modification only implies software but no hardware reprogramming. Work companion doubtlessly implies less cognitive loads than the home companion, working in known or dedicated environment with qualified surrounding humans.

Like for all mankind creativity process, universal grunt-soldier robots are of course another expected outcomes. The latest results of the United States of America, the Atlas humanoid robot, demonstrates impressive movement capabilities in its chemical suit [Nelson 12]. The quadruped Alpha Dog robot, is now training with the America military forces, while the most advanced humanoid robotics challenge is organized by the American defense advanced research projects agency, DARPA.

Telerobotics: Before the challenge of an autonomous humanoid robots, humanoid robotics can make a first coming-out with teleoperated semi-autonomous robots. As explained in the introduction, the humanoid shape greatly eases the transfer of motion from the human operator to the robot. The importance of the dynamics differences between the robot and the operator creates another challenge for motion replication. There is in fact a large and continuous range of applicative situations, from the basic automaton to the cognitive companion, emphasized by the DARPA humanoid challenge [Guizzo 12] and the challenges open with the disaster of Fukushima, in particular to visit then repair, clean and destroy the damaged plant [Guizzo 11].

Computer animation: Providing virtual humans with autonomy of motion is a long problem in computer animation [Pettré 08]. For long, in such problems, animation techniques were motion-capture based. The key idea of such techniques is rather simple: sequences of motion captures describe the ability of motion for digital creature. Motion planning is performed by assembling various portions of motion capture in order to reach a desired state (position, posture). Motion capture edition techniques (warping, blending or concatenation) are used to enable composed motions with smooth transitions and to adapt to the environment.

More recently, dynamic simulation has been used to control virtual humans. It overcomes the limitations imposed by motion capture based techniques. The problem has benefited from research in robotics. Seminal work from Koga and colleagues coupled motion planning techniques and animation techniques to enable autonomous motions [Koga 94]. This get animation problems very close to robotics one, even if constraints differ (natural aspect of motion is a preeminent target, and violating the law of Physics is doable). This work is taking much inspiration from publication in the computer animation field [Boulic 92, Baerlocher 04, de Lasa 10, Mordatch 12, Al Borno 13].

Biomechanics and neuroscience: The human body is *another* robot model that can be animated by just the same robotics methods. From the mechanical point of view, it is a very redundant kinematic tree with redundant actuators (each joint being actuated by several

muscles) and subject to kinematic or empirical constraints. The challenge is now to generate natural motions, for example for virtual ergonomics studies: the body model is positioned in a digital mock-up and the artificial movements are used to evaluate the potential occurrence of functional disorders [Fourquet 07].

Some aspects of the motion can be explained (and thus generated) by the kinematics of the body. Refining the motion is expected to improve the realism, by adding the inertia, the muscle positioning, the muscle activation and co-activation models, etc. These questions concern various sensorimotor aspects from biomechanics to neurosciences: how are disctributed the internal forces, what are the sensori-feedback loops, how is encoded the fusion of several sensing modalities (*e.g.* vestibular, vision and tactile) or of several simultaneous motor actions (*e.g.* reaching and balance, walking for reaching, etc.). In that challenge, robotics is bringing some modeling tools. The dialog between the two scientific corpus can also give an original enlightenment of robot decision process.

2.2 Original approach

Our work is based on local sensorimotor loops that enable the robot to robustly handle its environment. The task function approach, introduced at the end of the 80's, gives an unifying framework to control the motions of a robot in the real world, uncertain and changing. But at the same time, the task function approach provides a way to describe and then reason about the motion. The task is then a lexeme describing the movement and, at the same time, an effective controller.

The project presented in this document is built upon this observation and aims at developing effective motion-generation methods for various kinds of robots, and more particularly robots with a large number of degrees of freedom. The ambition of the approach is to provide computational solutions that, on the one hand, are not affected by the combinatorial explosion of the planning algorithms due to the large number of degrees of freedom and, on the other hand, are not affected during the execution by the underlying uncertainties of the interaction with the physical world.

This approach requires two complementary phases. We first establish the basic vocabulary of action, which will then be used to construct complex motions. The objective is to build an exhaustive control layer, covering all the working modalities of the robot, while providing a higher level with quasi symbolic control and diagnostic access. The core of this first phase is the definition of a stack-of-tasks structure. Based on this action vocabulary, the second phase of our research is to provide semantics of the movement, to ease all the aspects of the definition of a complex movement, *i.e.* motion programming, planning or learning.

We designed this approach in two phases, first with the construction of a motion vocabulary and second with the exploration of the corresponding semantics, by the generic term of motion semiotics.

2.3 Outline

The document is organized in two parts, following the bottom-up approach.

The first part tries to build a task-based controller that covers all the modalities of the robots. In **Chapter 3**, we recall the basis of the task-function approach and the limitations

that we are then trying to overpass. **Chapter 4** proposes two solutions to take into account inequality-written objectives in a hierarchy of tasks. **Chapter 5** describes the problem of the continuity during a sequence of tasks and proposes two preliminary solutions to this problem. **Chapter 6** then addresses the control of the dynamics of the humanoid robots. **Chapter 7** finishes the first part by relating the overall experimental works and in particular the need of sensory feedback.

The second part then uses the task-control paradigm built in the first part to assemble complex robot behavior. Three possibilities to build complex behaviors are explored: by programing in **Chapter 9**, by planning in **Chapter 10** and by learning from observation in **Chapter 11**. This last chapter opens to the study of natural human movements.

Finally, the conclusive **Chapter 13** opens to our perspectives and a more personal view of future humanoid robotics developments.

COD	Complete Orthogonal Decomposition
COM	Center of Mass
CRBA	Composite Rigid-Body Algorithm
CWE	Collaborative Working Environment
DOF	Degree of Freedom
FOV	Field of View
HCOD	Hierarchical Complete Orthogonal Decomposition
HQP	Hierarchic Quadratic Program
HPP	Humanoid Path Planner
HRP	Humanoid Robot Prototype
IMU	Inertial Measurement Unit
MPC	Model Predictive Control
PLM	Product Life-cycle Management
QP	Quadratic Program
RNEA	Recursive Newton Euler Algorithm
RRT	Rapidly-exploring Random Tree
SLAM	Simultaneous Localization and Mapping
SQP	Sequential Quadratic Program
SVD	Singular Value Decomposition
ZMP	Zero Moment Point

Table 2.1 *– List of abbreviations*

Part I

Action Vocabulary

Chapter 3

Task function

This chapter is built as a tutorial of existing solutions to implement the task-function approach for inverse kinematics. The robot configuration vector is denoted by q, and its temporal derivative $\dot{q}$.

3.1 Inverse geometry

3.1.1 Problem definition

A motion executed by a robot can be driven by a goal configuration q^*. However, the configuration space does not have in general a very intuitive structure. It is not expected that q^* is directly given by a human operator.

Direct and inverse geometry: Instead, the objective can be defined by means of a function of task $h(q)$, that should reach a given value h^*. The objective is not given in the configuration space by q^* but in the task space (the image space of the task function) by h^*. The function h is given as a direct map of the robot geometry and is here referred to as the direct geometry function. A typical example is to define h to be the robot end-effector translation, orientation or placement[1]. In that case, h is often referred to as the direct kinematics of the robot[2].

Analytical inversion: We first search one configuration q^* that satisfies the task h^*, *i.e.* such that $h(q^*) = h^*$. If h is invertible, the problem corresponds to finding the inverse map h^{-1}. Early works for six degree-of-freedom (DOF) manipulators use an analytical inversion

[1] The position of a body in space is defined by three translation and three rotation parameters. The placement refers to the concatenation of these two functions. It is typically defined by six parameters.

[2] The map that gives the position of the end effector with respect to the robot configuration is often called direct kinematics. The kinematics being the branch of mechanics that studies the *motion* ($\kappa\iota\nu\eta\mu\alpha$, motion in Greek) of sets of points, *i.e.* velocity, acceleration, we rather use direct geometry for the function h.

of the h map [Pieper 68]. A good overview of analytical methods is given in [Tolani 00] and [Waldron 08]. Analytical methods require a specific study for each function h, *i.e.* for each robot and task. The problem has been widely studied for specific shape of manipulator arms [Primrose 86, Lee 88] with finally some efficient methods proposed [Manocha 94]. However, the same study has to be realized for each new function h. For example, inverting the function that links the position of the robot to the position of some visual feature on the camera projection plane would require to do all the computation again. Moreover, h is not always invertible: the analytic inversion then does not systematically apply. In particular, when the robot is redundant with respect to the task, h is not injective, *i.e.* many solutions q gives a same h^*.

3.1.2 Numerical resolution

Instead, the problem of finding a configuration q where h^* is reached can be solved numerically. The problem is written as a least-square non-linear problem:

$$\min_q ||h(q) - h^*|| \tag{3.1}$$

We denote by $f : q \to f(q) = \frac{1}{2}||h(q) - h^*||$ the underlying scalar positive function. If h^* is in the range space of h, the minimum of f will be a root. Most of the time in robotics, h has a nice local behavior (local convexity, differentiability) coming from its geometrical structure. Differentiable iterative algorithms with descent directions are then well suited to solve it. Basically, these algorithms start with an initial guess q_0 (that should be such that $h(q_0)$ is not too far from h^*) and build a sequence of $q_1, q_2, ...$ that converges toward an optimum.

Gradient descent: At each iteration, the step is chosen to be in the opposite direction to the gradient of f at the current position:

$$q_{n+1} = q_n - \lambda \nabla f|_{q=q_n} = q_n - \lambda J(q_n)^T (h(q_n) - h^*) \tag{3.2}$$

where $J(q) = \frac{\partial h}{\partial q}$ is the Jacobian of h computed in q and λ is the step length. This second solution is named linear search and is costly, but speeds up the descent. Gradient descents are known to be have a very slow convergence: the residual at step n is typically bounded by $O(1/n)$.

Newton descent[3]: The function f to minimize is approximated by a first-order Taylor development:

$$f(q_n + \Delta q_n) = f(q_n) + \Delta q_n \left.\frac{\partial f}{\partial q}\right|_{q_n} \tag{3.3}$$

The next step Δq_n is chosen to nullify the approximation. Since $\frac{\partial f}{\partial q} = J(q)^T (h(q) - h^*)$, we have:

$$q_{n+1} = q_n + \Delta q_n = q_n + \lambda(q) J(q)^T (h(q) - h^*) \tag{3.4}$$

where $\lambda(q) = \frac{||h(q)-h^*||^2}{||J(q)^T(h(q)-h^*)||^2}$. This is just the gradient descent (3.2) with a varying λ descent length. The descent will stop at the zero of the function, or in the first encountered local minimum.

[3] or Newton-Raphson, cf. wikipedia for the complete story of the two men.

Gauss-Newton descent: To really search for the minimum of the function, the descent can search for the zero of its derivative. The Taylor development is then extended to the second order. The Hessian then appears. Gauss-Newton descent applies the above scheme but neglect the second order variations of the sum of squares. The descent step is then:

$$q_{n+1} = q_n - \lambda J(q)^T (J(q)J(q)^T)^{-1}(h(q) - h^*) \tag{3.5}$$

when $J(q)J(q)^T$ is invertible. In that case, the matrix product is equal to the pseudoinverse of $J(q)$ [BenIsrael 03] detailed below.

The convergence rate of the Gauss-Newton is much better than the gradient descent (residue bounded by $O(1/n^2)$). As previously, the step length λ can be chosen *a priori* or optimized by searching along the descent direction to improve the convergence speed.

Step length: Descent algorithms imply two steps at each iteration: first, choose the direction of the descent and, second, choose the length of the step along this direction. The length is given by λ in the above presentation. It can be constant or computed a priori from the cost residue. However, to diminish the number of step to convergence, a search along the line defined by the descent direction is generally performed, for example using a dichotomy search with an initial point defined by the (given) problem scaling factor. This search can be costly but significantly reduces the number of descent steps. Optimization algorithms typically take very large steps, while, in robotics, we are often interested by keeping quasi infinitesimal steps to obtain a good approximation of the continuous underlying trajectory.

3.2 Inverse kinematics

The iterative descent methods do not produce only one optimal configuration but also a sequence of configurations from the initial guess and leading to the goal h^*. A continuous trajectory is obtained by integrating with respect to the abscissa s the following differential equation:

$$\frac{\partial q}{\partial s} = -\lambda J(q(s))^+ (h(q(s)) - h^*) \tag{3.6}$$

starting from the initial configuration $q(0) = q_0$.

The Jacobian gives the direct (forward) kinematics of the task[4]:

$$\frac{\partial h}{\partial s} = J(q(s))\frac{\partial q}{\partial s} \tag{3.7}$$

The direct kinematics is linear with respect to the motion $\frac{\partial q}{\partial s}$ in the configuration space. At each configuration $q = q(s)$ of the path, (3.6) implies the inversion of J. This problem is called the inverse kinematics and is formally written by the following quadratic problem (QP):

$$\min_{\frac{\partial q}{\partial s}} ||J(q)\frac{\partial q}{\partial s} - \frac{\partial h}{\partial s}^*|| \tag{3.8}$$

[4]Equation (3.7) is often referred to as the forward differential kinematics.

with $\frac{\partial h}{\partial s}^* = -\lambda(h(q) - h^*)$. The Newton algorithm is then a sequence of QP and is referred to as sequential quadratic programming (SQP).

Physically, the abscissa s is most of the time variable t. Equation (3.6) is then the robot velocity in the configuration space:

$$\dot{q} = -\lambda J^+(h(q) - h^*) \tag{3.9}$$

The velocity $\dot{q}$ will be considered as the control input in the following.

3.3 The task-function formalism

The integration of the previous differential equation gives a motion $q(t)$ from q_0 to the target. The corresponding motion $h(t)$ in the task space is the solution of the differential equation:

$$\dot{h} = -\lambda(h(t) - h^*) \tag{3.10}$$

This corresponds to an exponential convergence of h toward h^* with a rate λ.

The task function approach [Samson 91] was introduced to formalize the system behavior in the neighborhood of the task completion h^*. Consider first a full-rank task function h, *i.e.* the map is differentiable and invertible in the neighborhood of any relevant points (J then exists and is also invertible). Given some properties of the function h around the target h^*, the task is said to be admissible, which ensures a good behavior of the robot [Samson 91]. The size of the neighborhood is difficult to determine in practice. The task-function approach then builds a trajectory $h^*(t)$ that leads to the ultimate goal h^* [Mezouar 02]. The admissibility property implicitly defines a tube in the configuration space around this trajectory, into which the robot control is straightforward. In practice, the neighborhood is very often empirically considered to be large enough and the tube is then reduced to its trivial final section [Espiau 92].

More generally, the task function approach ensures that any stability property of the reference vector field $\dot{h}^*$ in the task space will be locally kept when transforming it into the configuration space:

$$\dot{q} = J^+\dot{h}^* \tag{3.11}$$

The task is then defined by the task function $h : q \to h(q)$ and the reference vector field $\dot{h}^*$ in the task space. If the vector field converges to a fixed point h^*, then the control law (3.11) converges to a fixed configuration q^* such that $h(q^*) = h^*$. The iterative inverse-kinematics problem then solves the geometric problem to find $h(q) = h^*$.

Several vector fields can be considered. The most classical one is the proportional (3.10) tuned by the proportional gain λ. Constant gains lead to slow convergence. An adaptive gain λ typically grows with respect to the task residue:

$$\dot{h} = -\lambda(||h||)(h - h^*) \tag{3.12}$$

with $\lambda(x) = (\lambda_0 - \lambda_\infty)e^{\frac{-x}{\beta}} + \lambda_\infty$ defined by three parameters. Force impedance [Hogan 84] can be achieved by integrating once the following second-order differential equation

$$M\ddot{h}^* = -B\dot{h} - Kh + \phi \tag{3.13}$$

where M, B and K are the inertia, damping and stiffness set to the system and ϕ is a measured external disturbance (typically, a force measured by a sensor).

A target can be reached in final time by imposing the system to follow a given parameterized time function [de Lasa 10]. For example, in order to follow a second order polynomial reaching 0 at time T, the vector field explicitly depends on the time variable and is defined by:

$$\dot{h}^* = (1 - 2\frac{\Delta T}{T})\dot{h}_0 - 2\frac{\Delta T}{T^2}h_0 \tag{3.14}$$

where h_0 and $\dot{h}_0$ are the measured position and velocity in the task space and ΔT is the integration time step.

The task function itself can also depend on time through changes of the environment. Denoting by Ω the external universe (all the variables that are not the robot configuration) and $\dot{\Omega}$ its changing rate through time, the task function is $h(q, \Omega)$ and its evolution is:

$$\dot{h} = J\dot{q} + \frac{\partial h}{\partial \Omega}\dot{\Omega} \tag{3.15}$$

where $\frac{\partial h}{\partial \Omega}\dot{\Omega}$ is a inherent task drift that should be at best estimated and compensated:

$$\dot{q} = J^+(\dot{h}^* - \frac{\partial h}{\partial \Omega}\dot{\Omega}) \tag{3.16}$$

3.4 Redundancy

The task function h defines completely the configuration corresponding to h^* if it is invertible. If it is not invertible, it might be not surjective: in that case the task is not feasible (there is a conflict between the task objectives that cannot all be met at the same time) or it has redundant output (some part of the task are *automatically* achieved as soon as the other part is achieved). If it is not injective, it has a redundancy of configurations (several configurations correspond to the achievement of the task). In robotics, redundancy usually refers to this second case.

The task-function approach primarily requires the invertible task function to be also a local diffeomorphism around the regulation point (*i.e.* J exists and is invertible) in order to keep the stability of $\dot{h}^*$ in the configuration space. One of the big interest of the task-function approach is to reduce the task study, which is non linear, to the study of the spaces tangent to the configuration space (space of $\dot{q}$) and to the task space (space of $\dot{h}$) that have a linear structure and are linked by J (which carries all the local non linearity of the problem). Consider then from now the linear problem to find $\dot{q}$ that satisfies $\dot{h}^*$ at best:

$$\min_{\dot{q}} ||J\dot{q} - \dot{h}^*|| \tag{3.17}$$

Null space: When J is not invertible, its null-space (kernel) is denoted by Z. A variation of the configuration $\dot{q}$ in Z will not have any effect in the task space. If Z is integrable, the intuition is very easy to catch: the position of the robot in the corresponding space is not controlled, can drift or exponentially diverge. In any case, when J is not injective, the stability of the system cannot be ensured, even if the $\dot{h}^*$ vector field is stable.

In order to preserve the stability, a secondary task can be employed and projected in the null-space of J [Liégeois 77]. The set of all solutions to the problem (3.17) is given by:

$$\dot{q} = J^{+}\dot{h}^{*} + P\dot{q}_2 \tag{3.18}$$

where P is any projector into the null space of J (*i.e.* $PP = P$ and $JP = 0$, for example $P = ZZ^T$ where Z abusively denotes a basis of the null space) and $\dot{q}_2$ is any vector that generates the solution set when varying.

The parameter $\dot{q}_2$ can be used as a secondary input to achieve another objective without disturbing the first task. In [Samson 91], this is used to ensure the stability of the whole system. The input $\dot{q}_2$ is chosen as the gradient of a potential field h_2 function of q:

$$\dot{q} = J^{+}\dot{h}^{*} + P\nabla h_2 \tag{3.19}$$

This scheme has been widely used to take into account the robot constraints when performing the task. For example, in [Liégeois 77], the potential field is a bowl-shaped function that pushes the robot away from the joint limits.

Weighted inverse: The set of solutions can also be described by choosing a different norm on $\dot{q}$ when inverting J (see Appendix A.3):

$$\dot{q} = J^{\#W}\dot{h}^{*} \tag{3.20}$$

where W is a weight (positive definite) matrix that generates the solution set when varying. The relation with the null space is less intuitive than in (3.18). Weighted inverse have also been used to take into account the robot constraints (*e.g.* joint limits [Chang 95]). It was used in [Zanchettin 12] to enforce a cycle of configuration-space trajectories when performing a redundant cyclic task. Cyclicity will of course ensure the stability of the robot despite the redundancy of the task.

Worst to be noted: when the task is fully constraining (no redundancy, J is full column rank), then P is of course null and W has no effect in (3.20).

Secondary task: To avoid confusion, the main task is now explicitly indexed by e_1. The potential h_2 in (3.19) is directly written as a function of the configuration. As for the main task, it might be desirable to write the field in a dedicated task space. Given a secondary task function e_2 and its reference behavior $\dot{e}_2^*$ (which can be the gradient of a potential field in the task space of e_2), the development of (3.19) leads to:

$$\dot{q} = J_1^{+}\dot{e}_1^{*} + P_1 J_2^{+}\dot{e}_2^{*} \tag{3.21}$$

This form was emphasized in [Chiaverini 97]. It ensures that the first task is perfectly accomplished. The second task is in general not full performed (except in particular cases). However, this is enough to ensure the stabilization of the system if both $\dot{e}_1^*$ and $\dot{e}_2^*$ are stable and the matrix obtained by stacking J_1 and J_2 is full column rank. Moreover, even though $\dot{e}_2^*$ is not perfectly accomplished, this scheme asymptotically converges toward the attractor of $\dot{e}_2^*$ if the tasks are compatible, *i.e.* if the rank of the stack (J_1, J_2) is the sum of the ranks of J_1 and J_2.

3.5 Hierarchy

3.5.1 Two tasks

The second task is not perfectly accomplished because the DOF used by e_1 are not taken into account when inverting J_2: the scheme does not compensate for the motion realized by e_1 when solving e_2. Instead, it is possible to search for the optimal $\dot{q}$ that solves the second task while preserving the first one. This is written by a QP:

$$\min_{\dot{q}} ||J_2\dot{q} - \dot{e}_2^*|| \tag{3.22}$$

$$\text{subject to} \quad \dot{q} = J_1^+\dot{e}_1^* + P_1\dot{q}_2$$

This problem is trivially reformulated as the following unconstrained QP:

$$\min_{\dot{q}_2} ||J_2P_1\dot{q}_2 - \dot{e}_2^* + J_2J_1^+\dot{e}_1^*|| \tag{3.23}$$

The set of solutions is obtained by taking the pseudoinverse of J_2P_1 [Siciliano 91]:

$$\dot{q}_2 = (J_2P_1)^+(\dot{e}_2^* - J_2J_1^+\dot{e}_1^*) + \tilde{P}_2\dot{q}_3 \tag{3.24}$$

where $\tilde{P}_2$ is the projector into the null space of J_2P_1 and $\dot{q}_3$ is any vector that generates the solution set when varying. The complete control for the two tasks is obtained by replacing $\dot{q}_2$ in (3.18):

$$\dot{q} = J_1^+\dot{e}_1^* + (J_2P_1)^+(\dot{e}_2^* - J_2J_1^+\dot{e}_1^*) + P_2\dot{q}_3 \tag{3.25}$$

where $P_2 = P_1\tilde{P}_2$ can be shown to be the projector into the null space of $\begin{bmatrix} J_1 \\ J_2 \end{bmatrix}$ [Baerlocher 04].

3.5.2 Multiple tasks

The same scheme can be extended recursively to take into account any number of tasks:

$$\dot{q}_i = \dot{q}_{i-1} + (J_iP_{i-1})^+ (\dot{e}_i^* - J_i\dot{q}_{i-1}) \tag{3.26}$$

with $\dot{q}_0 = 0$ and P_i the projector into the stack of the i first levels.

3.5.3 Discussion

Singularities: A singularity happens when the rank of the task Jacobian decreases compared to its nominal rank. Consider two tasks e_1 and e_2: a singularity arises when the rank of the Jacobian of (e_1, e_2) decreases, *i.e.* when e_1 or e_2 is singular by themselves, or when each one is non singular but both together conflict and become singular. The first case is often called kinematic singularity while the second case is called algorithmic singularity [Chiaverini 97].

Robustness: The two schemes (3.21) and (3.25) are compared in [Chiaverini 97]. Basically, (3.21) is less efficient (e_2 is not totally accomplished) but is also stable and is robust to algorithmic singularities. In fact, (3.21) does not take into account the coupling between the two tasks. Taking into account this coupling increases the efficiency of the solution, but simultaneously lead to algorithmic singularities. The robust (3.21) is also cheaper to compute. In a sense, applying (3.21) for realizing two tasks is similar to applying the Jacobian transpose for realizing one task: it is robust and cheaper, but is less efficient since the coupling (between each task or between each component of the task) is not taken into account.

Weighted sum: However, both singularities are similar in the sense that the combined task (e_1, e_2) is singular. Moreover, outside of algorithmic singular region (when both tasks are compatible), (3.25) is equal to the solution obtained with the two tasks combined in only one. The combination can also be done while imposing different weights on each task: $e_w = (w_1 e_1, w_2 e_2)$. The resolution of the QP associated to e_w is given by the left-weighted inverse of $J = (J_1, J_2)$:

$$\dot{q} = J^{W\#} \begin{bmatrix} \dot{e}_1^* \\ \dot{e}_2^* \end{bmatrix} \tag{3.27}$$

with $W = diag(w_1, w_2)$ the weight matrix.

It can be shown that the hierarchy (3.25) is obtained when the ratio between the weights reaches the limit [VanLoan 85]: a strict hierarchy is obtained when the importance given to a task is infinite. Like the hierarchy, the weights are meaningless when the tasks are compatible.

The hierarchy can then be numerically obtained by using a single weighted QP with 10^3 weight ratios. This solution is often used for example on humanoid setups when using off-the-shelf solvers. However, the conditioning of the problem is then very poor. The usual solvers are often robust to such scaling problems, but, when possible, the use of a dedicated hierarchical solver avoids these artificial conditioning problems.

3.6 Challenges and objectives

The task function is a very good approach to drive a system toward the realization of a geometric objective. It is a local method, so it is not possible to expect a global solution. Moreover, it is only a constructive framework and the overall system behavior will depend on the choice of the selected task functions and of the reference vector fields in the chosen task spaces.

The task-function approach or similar (operational space control [Khatib 87], inverse kinematics and differential kinematics [Whitney 69], impedance control [Hogan 84], hierarchy of tasks [Baerlocher 04, 14], visual servoing [Espiau 92, Chaumette 06]) are widely used by the community and for real projects. Several aspects are still not properly treated and wait for a final solution.

Regularization: The inversion process inherently suffers from the sensibility to the system conditioning close to singular points. This aspect is well understood and proper solutions exist to avoid the singular region [Yoshikawa 84, Nelson 95, Marani 02] or to smooth the robot behavior when it is entering the region [Wampler 86, Deo 92, Sugihara 11]. However, the classical solution decreases the performances of the control by adding a damper. The

regularization is then invasive and strongly relies on a tradeoff between performance and safety. Of course, the parameter tuning is then context dependent and no good solution other than empiricism exists to choose it. More generally, there exists no generic solution to automatically regularize the motion problem while preserving the performances away from the singular region.

Task Composition: Among the generic regularization problems, two specific problems can be isolated: How to execute simultaneously several tasks (algorithmic singularity problem)? And how to sequence a set of tasks (task sequencing problem)?

As discussed above, the first problem would be directly solved if a complete regularization was available. Since it is not, the complexity is reduced by considering that all the tasks are non singular by themselves but that the problem is coming from the conflicts when composing them in a single motion [Chiaverini 97]. In that sense, imposing a strict hierarchy is a way to circumvent the composition problem: indeed, each task being decoupled from the others, the conflict will only produce an effect on one task. The consequences are then clearer to measure, analyze, and hopefully treat. Yet it could be hopped to treat the conflict for several tasks only at the level of one task, which would reduce the complexity. However, hierarchy is not yet a solution by itself, but just a way to isolate the problem and help to correct it. A complete solution to compose a set of valid tasks while avoiding the problems due to the conflicts is still missing.

When sequencing two tasks, or in a hierarchy when adding or removing a task from a running set, a discontinuity of $\dot{q}$ typically arises. This problem will be reduced to the regularization problem [Salini 09, 8] in Chapter 4. Indeed, a classical solution to smooth the transition between two sets of tasks is to use the damped inverse. Another solution to smooth the velocity is to move the whole system to the second order [Siciliano 90]. However the acceleration are then discontinuous, which is less problematic but should be avoided if possible. Two problematic cases have to be considered. If the robot behavior or structure is optimized by a solver [34], the discontinuity of the control may typically break the solver convergence. And if the sequence is triggered by an event rather than by a time sequence, the discontinuity may trigger another sequence and cause the system to loop around the discontinuity [11].

Inequalities: All the work above deals with equality objectives, typically trying to bring an error to 0 and the geometric map h to a desired point h^*. Many robotics objectives would rather be written in terms of inequalities: joint limits [Liégeois 77], field of view (FOV) [Marchand 98], obstacle avoidance [Khatib 86], actuator (velocity or torque) limits, safety region [Cheah 07], center of mass (COM) inside the support polygon [Sugihara 02], etc. Two classical solutions are used to handle inequalities. If the robot is redundant with respect to the task, the inequality can be embedded into a potential function (*e.g.* a log-barrier function [Nemirovski 94]), whose gradient is projected in the remaining null space. The inequality is never taken into account as an objective having priority [Liégeois 77, Marchand 98, Chaumette 01, Sian 05]; or a switching control law can be built, which sequentially moves from the inequality criteria when close to the saturation, to the driving task otherwise. Several ad-hoc developments have been tried in this second direction [Peinado 05, Raunhardt 07, Sentis 07, 14, 12] with problems of genericity; continuity during the sequencing; stability of

the switch [Gans 07]; and locking of the constraint [Sentis 07]. None of these two sets of methods are completely satisfactory.

And the robot? The computed $\dot{q}$ can be integrated to generate a motion, for animation [Boulic 92, de Lasa 10], virtual prototyping in product life-cycle management (PLM) [Weyrich 99, Laumond 06] or to be replayed in open loop by a robot. However, $\dot{q}$ can also be seen as a control law to be applied in real time on a real robot. In that case, some problems of computation time and of linkage with the measured state of the robot can arise and should be considered. In particular, the inverse kinematics only considers (by definition) the geometry of the robot and not the dynamics. The extension to inverse dynamics [Khatib 87] is possible, but there is no *good* solution yet to execute it with a robot [Whitney 76, Hogan 88]. Several solutions are currently explored: estimation of the whole dynamic model (with friction model in the actuators) [Khatib 08] ; a force sensor can be added to close the loop at the torque levels [AlbuSchaffer 07a] ; or spring mechanisms can be added to mechanically transform the torque control problem to a position control problem [Pratt 95, Kaneko 94]. Once more, due to calibration, price or control problems, none of these solutions are completely satisfactory.

The objective of this first part of the document (*i.e.* Chapters 4, 5, 6 and 7) is to set up a methodology to execute any set of tasks on a physical robot. The complete motion generator should tackle objectives written as equalities or inequalities function of the robot dynamics, and be able to compose any set of tasks, both hierarchically or sequentially, in a safe way. Hierarchy of both equalities and inequalities are proposed in Chapter 4. The problem of the continuous sequencing is properly defined in Chapter 5, and a first solution is proposed. Some first steps toward the application of hierarchy of tasks to inverse dynamics is then proposed in Chapter 6. Finally, the work done to close the loop with the sensors on the real robot is presented in Chapter 7.

Chapter 4

Inequality

The goal of this chapter is to define a way to handle inequality objectives into a hierarchy of tasks. Consider a main task driving the robot and an inequality constraint written in a second task space. Three main approaches to realize these two objectives can be found in the literature.

Potential field: The inequality can be embedded in a potential function [Liégeois 77, Khatib 86], like a log-barrier [Nemirovski 94] (see [Mezouar 02] for an example). The potential field can be directly taken into account in the task-function approach by projecting its gradient in the null-space of the other tasks. The gradient is a well-defined (stable) vector field in the task space. It acts as a virtual force [Khatib 86] that pushes the robot away from the forbidden areas.

The task enforcing the inequality constraints corresponds to some DOF of the system. These DOF are devoted to the inequality constraint even when the corresponding gradient is null. It is not possible to use them to fulfill an objective of lower priority. For example, when the inequality constraint represents the joint range of the robot, the corresponding task space is the configuration space whose null space reduced to zero: no other task can be added in lower priority and the joint limits cannot be considered as a top-priority objective. The gradient is generally taken into account as the least-priority objective [Marchand 98]. It is then only applicable if the robot is redundant with respect to the main task (and if possible if there is many degrees of redundancy).

Typically, if a seven-DOF manipulator robot is executing a translation task (three DOF), the remaining four DOF can be used to stay away from the joint limits. However, even in this case, no guarantee can be given on the avoidance. For example, on an anthropomorphic arm, if a limit prevents the elbow from completely stretching and the task objective is outside of the reachable region, the limit will be violated despite the potential field. Some other variations have been proposed, like [Nelson 95] that uses a trade-off between the task and the constraints, with no guarantee of avoidance.

In [Chang 95], the potential field is used to weight the pseudoinverse rather than to project

a virtual force. The weight tends toward infinity close to the constraint limit (joint limits in the paper), which forbid further motion. However, when several joints are hit, the solution reaches a kind of singularity with the inversion becoming invariant to the weight. Here also, avoidance cannot be ensured. This solution is interesting because it is an intermediate stage between clamping and switching control: the method pushes the weight toward the limit, but never reaches it. When the limit is reached, it clamps the robot motion in the constraint direction.

Switching control: The potential field cannot be set at the top priority because all the components of the inequality task are always active: even far from the inequality border, when the gradient is null, the DOF is constrained and no other command can act on it. A solution is then to artificially deactivate the constraint when far enough from the obstacle. Clamping was proposed to apply such an activation to enforce the joint limits [Raunhardt 07]. Clamping is easy: it is only a test over the robot preview window. If the constraint is violated, the controller clamps the direction to prevent any further motion. However, relaxing the constraint is more difficult. An attempt is proposed in [Sentis 07] (Chapter 5). It is possible to find some observers that properly indicate when to remove one constraint. However, when several constraints are active at the same time and are coupled together, such observers are much trickier to build. Moreover, the stability of the system does not arise immediately from the task-function approach, but the switching state machine should also be considered in the stability loop. The inability to prove the stability of such a system can lead to very serious problem during the robot execution. Typical examples of oscillations are shown in detail in [11].

During my PhD, we proposed a controller to guarantee that the potential field (set as a least-priority objective) is always taken into account during the realization of the main task [14]. For that purpose, a switching controller was in charge to deactivate parts of the main task if the robot was approaching to close to the limit. This can be seen as a clamping controller. However, instead of activating a part of the constraint, the controller deactivates the corresponding part of the main task so that the potential field carrying the constraint can be properly taken into account. This approach reduces the complexity to decide which part of the constraint was not necessary any more. However, several aspects of this development were dedicated to the considered tasks, constraints and robot structures and would need specific development to extend to other situations.

QP-based control: It was shown in the previous chapter that the pseudoinverse control is the result of a unconstrained QP minimizing the distance to the reference task vector. QP can also be defined with inequality or equality constraints [Nenchev 89, Sung 96, Decré 09]. In that case, the inequality can be handled at the top priority level by setting it in the constraint of the QP. It is also possible to take into account inequalities at the second level by introducing a slack variable [Boyd 04], as it was done in [Hofmann 09] to control the balance of an anthropomorphic avatar. Similarly to the case of unconstrained QP that was linked to unconstrained SQP, inequality-constrained QP can be linked to inequality-constrained SQP.

A QP solver is composed of a constraint and a cost, which can be seen as two levels of constraint having different priorities. The approaches cited above are unable to take into account a hierarchy of more than two objectives. A hierarchy-like behaviors is achieved by using some proper weights to set up the relative importance of the various objectives

[Collette 07, Liu 11]. However, as explained before, the conditioning of the weighted task is poor and can lead to artificial numerical during the problem resolution.

Two main approaches have been tried. First, it was tried to extend the work of [Chang 95] and to actually clamped the problematic constraint directions when coming to close to the limit. This was done by redefining an inversion operator whose behavior is close to the weighted inverse in the good cases, but that ensures a smooth behavior when crossing the singularity due to the non-definiteness of the weight matrix. The resulting behavior of the system is very good, but the cost of the operator is too high to apply to realistic cases with a humanoid robot.

Concurrently, the QP-based approach was extended to take into account a hierarchy of objectives. The behavior is more intuitive and the resolution is much cheaper. However, there is no good solution yet to damp the problem and ensures a safe robot behavior close to the singular regions.

These two approaches will be detailed in the next two sections.

4.1 Multi-Dimensional Homotopy

Reference paper [12] (see also [11, 40, 37, 57])

4.1.1 Considering one task

The starting idea is to enable at the top-priority level the activation and deactivation of parts of the task, that are used to clamp or relax the DOF approaching to the constraint limit. We consider a task function $(e, J, \dot{e}^*)$, each of whose component is parameterized by an activation factor h continuously varying from 0 (fully inactive) to 1 (fully active). A common solution to take into account such a task is to consider the task $e_H = He$, where $H = diag(h)$ is the activation matrix. Neglecting the variation of H, the Jacobian of e_H is HJ and the resulting control law is

$$\dot{q} = (HJ)^+ H\dot{e}^* \tag{4.1}$$

The shape can be found *e.g.* in [Cheah 05, Comport 06, GarciaAracil 05, Chang 95].

We can recognize here the left-weighted inverse of J: $\dot{q} = J^{H\#}e$. As recalled in Appendix A.3, the weighted inverse is only properly defined when the weight matrix is positive definite and is invariant to the weight when the Jacobian is full-row rank. Consequently, the control law (4.1) behaves improperly when one component of H becomes null (H becomes non definite) and when subparts of J are full row rank and *decoupled* from the rest of the Jacobian. The problems come from a singularity due to the matrix shapes that arises from the computation of $J^{H\#}$. Consequently, discontinuities happen when a component of e becomes fully inactive. The control law (4.1) that seems to be continuous in fact behaves like a switching control law and can be put in oscillatory modes with very simple setups [57].

To avoid these singular behaviors, a new inversion operator $.^{H\oplus}$ is defined that behaves similarly to $.^{H\#}$ far from the singular regions but ensures the continuity during the transition. The operator is obtained by inverting all the row submatrices of J (matrices composed of some rows of J) and summing them with weights depending on the corresponding components of H. When a single task is considered, the control is:

$$\dot{q} = J^{H\oplus}\dot{e}^* \tag{4.2}$$

It is proved continuous and stable in [11]. It can be seen as a homotopy from the unconstrained control law to the control law clamped by the constraint. When several constraints are considered simultaneously, it is a multiple homotopy that gives more relative importance to the closer constraints.

4.1.2 Considering two tasks

When a second task is considered, it can be simply projected into the corresponding pseudo null space obtained with $P_\oplus = I - J^{H\oplus}J$, following the scheme (3.21):

$$\dot{q} = J^{H\oplus}\dot{e}^* + P_\oplus\dot{q}_2 \tag{4.3}$$

with for example $\dot{q}_2 = J_2^+ e_2$ realized a second task is $(e_2, J_2, \dot{e}_2^*)$. Similarly to (3.21), the secondary task is not optimally executed since $P_\oplus$ is not taken into account in the inversion of J_2. Again by analogy with (3.25), the pseudo projector can be handle in the inverse by:

$$\dot{q} = J^{H\oplus}\dot{e}^* + P_\oplus(J_2P_\oplus)^+\dot{e}_2^* \tag{4.4}$$

As before, the second matrix can be identified as the right weighted inverse $J_2^{\#P_\oplus}$. As before, the inversion behaves improperly when $P_\oplus$ loses its definiteness, *i.e.* when one component of e becomes fully inactive. As before, the operator $.^\oplus$ is extended to handle right-weighting while keeping a behavior similar to the weighted inverse. The control law for two tasks can be written:

$$\dot{q} = J^{H\oplus}\dot{e}^* + J_2^{\oplus P_\oplus}\dot{e}_2^* \tag{4.5}$$

The same method can be extended to a hierarchy of several tasks. See [12] for details.

4.1.3 Implementation for avoidance

An inequality constraint $e > 0$ can be implemented by setting for $\dot{e}^*$ a vector field that asymmetrically pushes the robot:

$$\dot{e}^* = \begin{cases} -\lambda e & \text{if } e < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

The activation matrix H is 1 when the constraint is violated, 0 far from the constraint limit and continuously evolves from 1 to 0 in an activation buffer defined on the right neighborhood of the constraint. The components of H can typically be linear (C^0 continuity), polynomials or chosen to ensure a C^∞ continuity (see [11]).

The control (4.5) has then two effects: on the one hand, it pushes the robot away from the constraint limit. On the other hand, it progressively clamps the motion toward the limit when close enough. In that sense, it combines the effects of both the projected gradient [Liégeois 77] and of the clamping [Raunhardt 07] and therefore prevent the constraint from being violated while automatically relaxing the constraint when possible. The system stops the robot motion before the exact limit, while performing a tradeoff between the clamping effect tuned by the slope of H and the pushing effect tuned by the gain λ.

In [12], the same approach is applied to the inverse dynamics in the operational space. It corresponds to a simple extension to another quadratic problem. The links between inverse kinematics and dynamics is the subject of Chapter 6.

4.1.4 Results

The need for a continuous behavior at the transition clearly appears in the toy example summarized in Fig. 4.1. The control law was then applied to the control of the visibility during a visual-servoing task [40, 11] and to avoid the joint limits while positioning the eye-in-hand robot arm in front of a target [37] as shown in Fig. 4.2. A more complex hierarchy is also used to control the hand of a humanoid avatar while preserving a natural posture [12].

4.1.5 Conclusion and future work

The scheme provides a very good robot behavior: there are few parameters to tune and the control is not very sensitive to improper tuning. The robot behavior is smooth and predictable. It is easy to damp ill-conditioned situations using the classical damped inverse instead of the pseudoinverse in all the computations.

There are two limitations that prevent the use of this scheme as a final solution. First, the activation-deactivation H is function of the robot configuration[1]. It is not possible to set it as a function of the control: for example, it is not possible to impose velocity limits in inverse kinematics, or torque limits in dynamics. Second, the continuous inverse $.^{\oplus}$ requires to compute the inverse for all possible sub systems. The cost is then exponentially growing with the problem dimension. For animating a robot arm, the cost is acceptable. However, for a humanoid, it is quickly too expensive for a real application. It does not seem possible to lower this cost with the same operator. Preliminary works have been done in [8] to try to approximate $.^{\oplus}$ and break the exponential complexity. This will be discussed in Chapter 5.

4.2 Hierarchical Quadratic Programing

Reference paper [4] (see also [28])

4.2.1 Cascade of QP

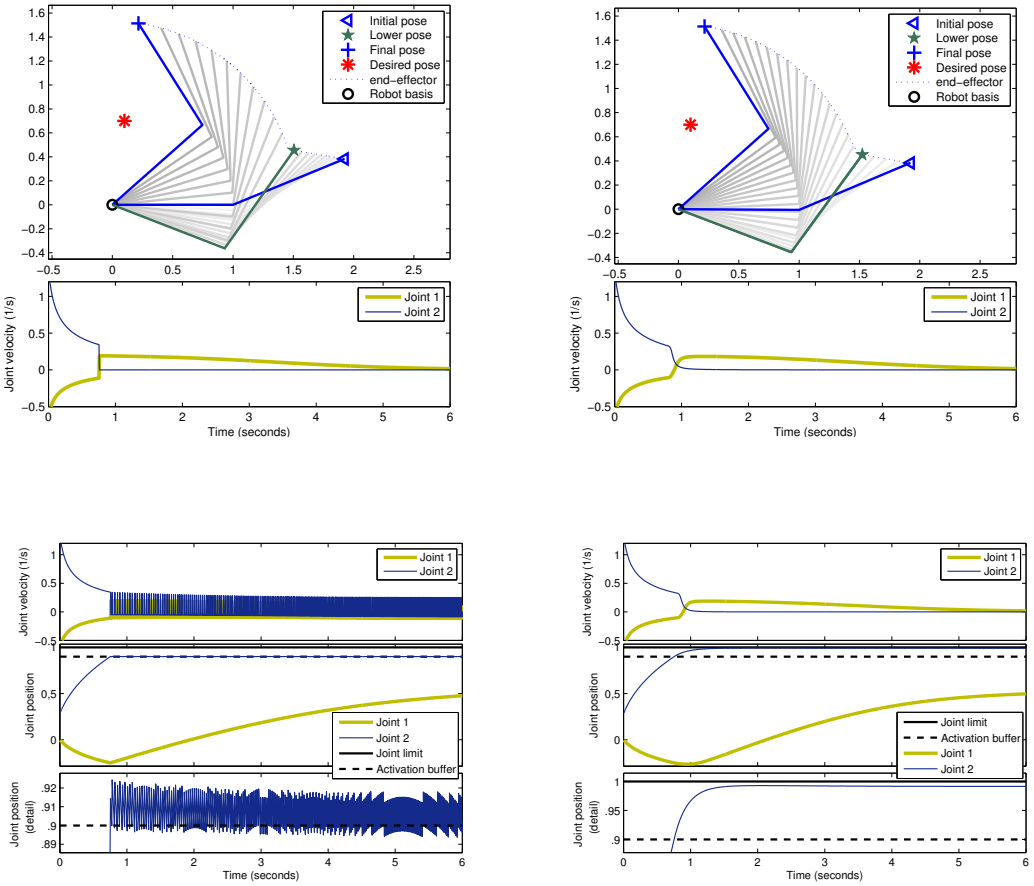
We have seen in Chapter 3 that the inverse kinematics problem can be written as a unconstrained QP. Similarly, the hierarchy of task can be written using a cascade of QP [Kanoun 11a]. Using generic notations[2], consider a variable x and a hierarchy of p levels defined each by a matrix A_i and a target vector b_i (typically in inverse kinematics, $x = \dot{q}$, $A_i = J_i$ and $b_i = \dot{e}_i^*$). At the first level $i = 1$, the QP to solve is simply:

$$\min_x ||A_1 x - b_1|| \tag{4.7}$$

An inequality objective can also be considered by introducing a slack variable [Boyd 04, Hofmann 09, Kanoun 11a]:

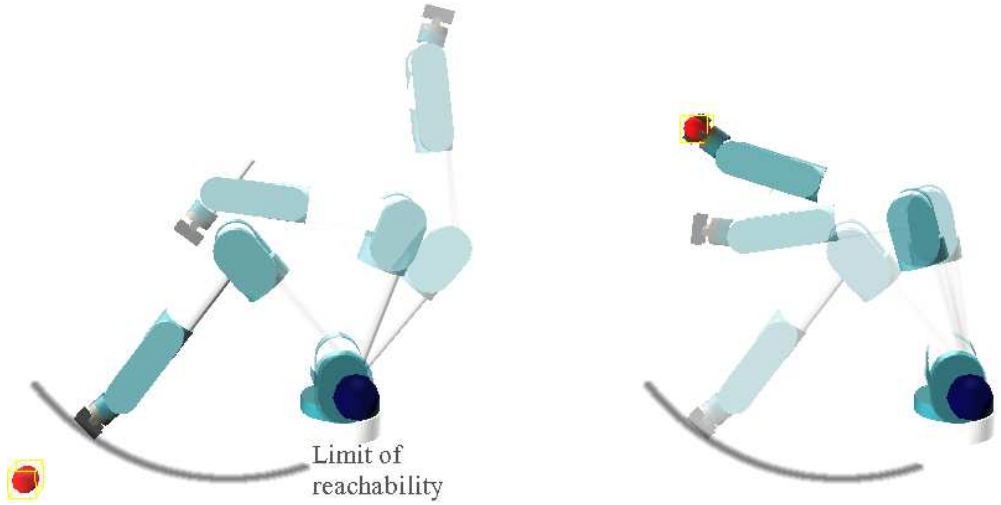
[1] If moving to the second order (dynamics), it can also be function of the velocity

[2] The generic solver will then be used to compute the inverse kinematics ($x = \dot{q}$) and the inverse dynamics ($x = (\ddot{q}, \tau, f)$, see Chapter 6).



Discontinuous control Continuous control

Figure 4.1 – *Positioning a toy 2D robot arm while avoiding the joint limits. The robot has to reach the star, while a limit prevents the elbow from bending. On the left, the control is discontinuous. When the gain λ of the avoidance is null (top figure), the discontinuity is not critic. It leads to an oscillatory behavior on the activation border as soon as λ is not null (bottom behavior). On the right column, the continuous control law is used: the control is smooth and no oscillation appears even when λ is not null. The robot does not exactly stops on the limit but a little bit ahead. The smoothed switching behavior at $t = 1s$ clearly appears on the velocity plot.*



(1) (2)

Figure 4.2 *– Reaching an object behind the limit of the accessibility domain. The joint-limit task has priority over the grasping task. The control prevents the violation of the limit by stopping the positioning task when the target is too far from the robot. When the robot moves to a second target, the constraint is properly relaxed.*

$$\min_{x,w_1} ||w_1|| \tag{4.8}$$
$$\text{subject to} \quad A_1 x \leq b_1 + w_1$$

The optimal slack w_1^* will correspond to the least-square residue. If possible, it will be set to 0 by the solver. A second level of constraints can now be introduced. To enforce the hierarchy, the slack of the first level is set constant to the optimum w_1^* while the slack of the second level can vary. Iteratively, the QP of level i is defined by:

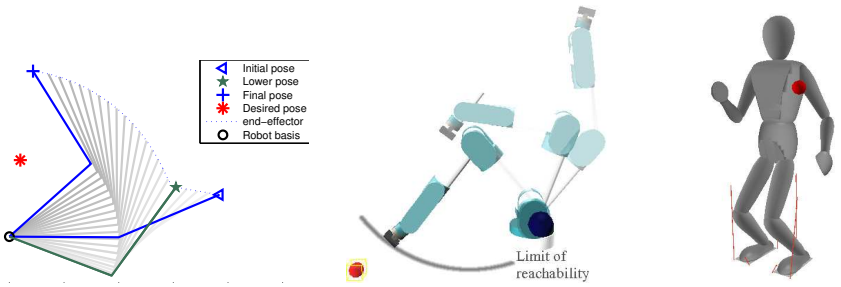
$$\min_{x,w_i} ||w_i|| \tag{4.9}$$
$$\text{subject to} \quad \begin{aligned} \underline{A}_{i-1} x &\leq \underline{b}_{i-1} + \underline{w}^*_{i-1} \\ A_i x &\leq b_i + w_i \end{aligned}$$

with the generic notation $\underline{X}_i = \begin{bmatrix} X_1 \\ \vdots \\ X_i \end{bmatrix}$.

The w_1 ... w_p and each x_1 ... x_{p-1} are internal variables. We are only interested by the computation of x_p^* which is the optimum for the whole hierarchy. The cascade defines a hierarchical quadratic program (HQP), denoted by:

A unified approach to integrate unilateral constraints in the stack of tasks

N. Mansard, O. Khatib, A. Khedar



IEEE Trans. on Robotics, 2009 [12]

Context:
This paper was written during my post-doctoral stay in Stanford, at O. Khatib's Laboratory. It is the generalization for multiple tasks of the work started with F. Chaumette and A. Remazeilles at INRIA during my PhD and published in [11].

Motivations:
Consider *e.g.* the joint limits. Far from a limit, the constraint should not be taken into account to keep as many DOF as possible for the other tasks. When approaching to the limit, the constraint has to be considered at the top priority to prevent the collision. One solution is to consider the limit constraint as a control feature that can be active or inactive. A discontinuity appears at the activation, which is often smoothed by using an approximation of the control, for example a damped inverse. The paper proposes a solution with better properties to handle such varying-set features in a hierarchy of tasks.

Approach:
With one task, the solution consists in making a weighted sum between the control laws with and without the active constraint. The weight modification drives a homotopy from the active to the inactive states. The solution can be summarized into a matrix form, by introducing a continuous-inverse operator, that is the homotopy of the pseudo-inverses of the various considered activation states. The continuous operator is then generalized to handle the redundancy projector and finally the hierarchy of tasks in inverse dynamics.

Results and contributions:
A stack of dynamic tasks considering various constraints has been built and applied to several setups (manipulator robots, anthropomorphic manikins). The obtained control law is very smooth and easy to implement and apply.

Limitations and perspectives:
When many activations occur at the same time, the complexity grows exponentially. It does not seem possible to find any alternative formulations to handle this problem. The approach is very satisfying in terms of control behavior, but is limited to simple cases.

$$\underset{x}{lsprox}\,(4.10\#1) \prec (4.10\#2) \prec \cdots \prec (4.10\#\mathrm{p}) \tag{4.10}$$

$$\begin{aligned} \text{subject to} \quad & A_1 x \leq b_1 && (4.10\#1)\\ & A_2 x \leq b_2 && (4.10\#2)\\ & \vdots \\ & A_p x \leq b_p && (4.10\#\mathrm{p}) \end{aligned}$$

where *lsprox* is a shortcut for *least-square approximation* and $\prec$ refers to the lexicographic order that imposes a hierarchy in the summation of the cost between the p levels.

A naive solution to solve the hierarchy is to execute each of the p QP. This is expensive for two reasons. The level i is first solved in the i^{th} QP but also in each QP of the following levels. For example, level 1 is solved p times, with the inverse of A_1 being computed each time. Additionally, each QP typically involves an iterative process used to solve the inequality constraints. As it will be detailed in Section 4.2.3, this iterative process badly interferes with the iteration along the cascade of QP and artificially slow down the resolution.

In the following, we will quickly summarize the work done to construct a dedicated HQP solver based on a primal active-search algorithm. If the developments are specific to quadratic problems, the approach is generic and could be applied to other class of optimization problems (conic, non-linear SQP) to make them *hierarchic*.

4.2.2 Equality-only hierarchical quadratic program

Decomposition of one level: When only equality constraints are considered, the HQP solution is nothing more than the *classical* robotics null-space hierarchy (3.26) recalled here:

$$x_i = x_{i-1} + (A_i P_{i-1})^+ (b_i - A_i x_{i-1}) \tag{4.11}$$

with P_i the projector into the null space of $\underline{A}_i$. Most of the time, each pseudoinverse is computed using a complete decomposition of the Jacobian matrices, typically a singular-value decomposition (SVD) [BenIsrael 03] or, a little bit more efficient, a complete orthogonal decomposition (COD) [Golub 96]:

$$A = \begin{bmatrix} V & U \end{bmatrix} \begin{bmatrix} 0 & 0 \\ L & 0 \end{bmatrix} \begin{bmatrix} Y & Z \end{bmatrix}^T \tag{4.12}$$

where $\begin{bmatrix} Y & Z \end{bmatrix}$ and $\begin{bmatrix} U & V \end{bmatrix}$ are two bases of respectively the input and the output space and L is a square invertible matrix that carries all the information of A. It is easily deduced that Z is a basis of the null space of A while U is a basis of the range space. A SVD will produce a diagonal L (whose components are the singular values) while a COD produces a triangular L. The pseudoinverse of A is then:

$$A^+ = \begin{bmatrix} Y & Z \end{bmatrix} \begin{bmatrix} 0 & 0 \\ L^{-1} & 0 \end{bmatrix} \begin{bmatrix} V & U \end{bmatrix}^T = Y L^{-1} U^T \tag{4.13}$$

Decomposition of several levels: We have defined a decomposition associated to the hierarchical structure of the problem, named hierarchical complete orthogonal decomposition (HCOD). It is defined by:

$$\begin{bmatrix} A_1 \\ \vdots \\ A_p \end{bmatrix} = \begin{bmatrix} W_1 & & \\ & \ddots & \\ & & W_p \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ L_1 & 0 & 0 & 0 & 0 \\ \boxed{\begin{matrix} N_2 \\ M_2 \end{matrix}} & L_2 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ \boxed{\begin{matrix} N_p \\ M_p \end{matrix}} & & & 0 & 0 \\ & & & L_p & 0 \end{bmatrix} \begin{bmatrix} Y_1 & \dots Y_p & Z_p \end{bmatrix}^T \tag{4.14}$$

where each $W_i = \begin{bmatrix} V_i & U_i \end{bmatrix}$ is a basis of the output space of each A_i, $Y = \begin{bmatrix} Y_1 & \dots Y_p & Z_p \end{bmatrix}$ is a basis of the input space such that $\begin{bmatrix} Y_{i+1} & \dots & Y_p & Z_p \end{bmatrix}$ is a basis of the null space of $\underline{A}_i$ (and then Z_p is a basis of the remaining null space of the hierarchy) and each L_i is triangular invertible.

Intuition: Each Y_i describes the DOF that can be used to satisfy level i. Z_p are the remaining DOF that are not used by any level.

The terms $\begin{bmatrix} N_i \\ M_i \end{bmatrix}$ represents the couplings between the level i and all the previous levels. More specifically, the row N_i, which corresponds to a null component on the column corresponding to Y_i, is the part of the task that cannot be accomplished. If N_i is null, this singularity is not coupled to any other task: it is a kinematic singularity [Chiaverini 97], *i.e.* a singularity that is due to the intrinsic structure of the task. If N_i is not null, there is a coupling with upper-priority tasks that explains the singularity: the singularity is algorithmic [Chiaverini 97], *i.e.* is due to a conflict with the hierarchy.

For each level, W_i partitioned the output space in two: U_i is a basis of the range space, *i.e.* the accessible space. V_i is a basis of the space that cannot be reached, and the corresponding part of the task $V_i^T b_i$ will not be performed.

Inversion: The HCOD emphasizes the hierarchical structure of the problem. Solving the hierarchy is now simply a matter of inverting the L_i. The solution to the HQP is given by:

$$x^* = \begin{bmatrix} Y_1 & \dots & Y_p \end{bmatrix} \underline{y}_p^* \tag{4.15}$$

where $\underline{y}_p^*$ is recursively defined:

$$\underline{y}_i^* = \begin{bmatrix} \underline{y}_{i-1}^* \\ L_i^{-1}(U_i^T b_i - M_i \underline{y}_{i-1}^*) \end{bmatrix} \tag{4.16}$$

with $\underline{y}_0$ the empty vector.

Comparing (4.16) to (4.11), $L_i^{-1} U_i^T$ stands for $(A_i P_{i-1})^+$ while $M_i \underline{y}_{i-1}$ stands for $J_i x_{i-1}$. The combination was done using a sum in (4.11) while each level is stacked in (4.16) (the sum then comes from the multiplication with the Y basis).

The HCOD is a decomposition that arises naturally when considering the iterative decompositions to compute the optimum of a HQP. The optimum is then simply computed by inverting the core of the decomposition. The optimum is finally computed using a scheme that, behind the appearances, is very similar to the classical one.

The improved performances are achieved by avoiding to explicitly compute the projectors at each level and by keeping the same basis Y to execute all the computations of the inverse (4.16).

Moreover, the structure of both the HCOD and the optimum (4.16) are interesting by the fact that they emphasize the internal structure of the problem. For example, the N_i reveal the internal couplings, the U_i, V_i separation gives an understanding of the infeasible part of the task, while the y_i^* indicate which level is asking for some high value of the optimum. This structure can be used to understand a particular HQP and to make some automatic diagnostic on the system.

4.2.3 Inequality hierarchical quadratic program

Active-search algorithm is a class of algorithms to solve a QP subject to inequality constraints. It tries to guess which subset of the inequality constraints hold as equalities at the optimum. This subset is called the active set. The algorithm starts with an initial guess of the active set. At each iteration, it solves the equality-only QP obtained by considering only the active constraints as equalities. From the solution obtained from this sub QP, a modification is done on the active set. The algorithm then iterates until the optimal active search is found, for which the associated equality QP gives the optimum of the inequality QP. Proof of convergence of the active-search loop can be given [Boyd 04].

We have adapted the classical active-search algorithm to solve the HQP (4.10). The algorithm works similarly, by computing at each iteration the optimum of the associated equality-only HQP using (4.15). The Lagrange multipliers, which correspond to the optimum of the dual problem, also have to be computed at each iteration. A proof of convergence of the active-search loop is given by construction. An alternative proof was proposed by P-B. Wieber using the lexicographic inherent structure of the hierarchy, based on the work of [Isermann 82] for hierarchical linear programs.

4.2.4 Properties and results

The cost of the equality-only HQP resolution is about n^3 with n the dimension of the variable x, that is to say similar to the cost of a QP of the same size. The HQP resolution can also be shown to be continuous with respect to the evolution of problem definition A_i, b_i, outside of the singular regions. Finally, the resulting control scheme in inverse kinematics can be shown stable despite the activation of constraints and asymptotically stable when enough DOF are available (no surprise indeed).

Comparison between the equality-only HQP solver (4.15) and the classical (4.11) have been done. A rough summary is given in Fig. 4.3. More details are available in [4]. As expected, the classical (4.11) is slower. More surprising, for problems of similar size, the cost increases when few levels are set. This can be understood when looking at the structure of the HCOD: to obtain a full COD (or when one single level is set), the N_i should be converted to 0 using additional transformations in Y, which increases the cost.

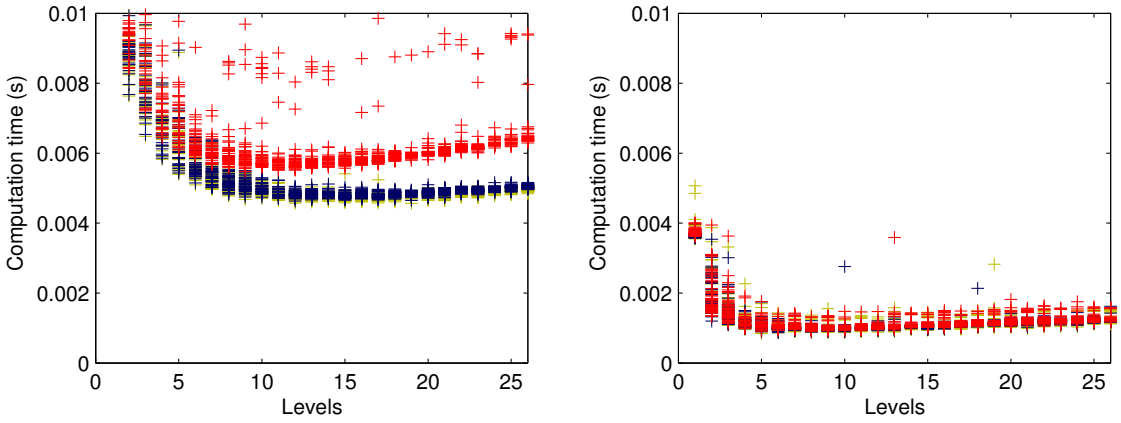


Figure 4.3 *– Comparison of the cost using a SVD and (4.11) (left) and a HCOD (right) to invert an equality-only HQP. The measures are done on a random set of HQP problems, where the size (number of rows, columns and total rank) are constant but where the number p of levels differs. The cost is plotted with respect to p. In red is the total cost, while blue is the cost spent in the decomposition computation and in yellow (behind the blue dots in the left plot) is the inversion. The HCOD is up to five times faster. Moreover, for both methods, the cost increases for small p.*

A similar comparison in the case of inequalities has been done between cascades of QP and HQP in Fig. 4.4. The cascade structure disturbs the active-search loop, that has to activate and inactivate several times the same constraint before converging, while the hierarchical active search is not disturb. The small overhead when p increases might be due to implementation problems or increased cost due to the computation of the Lagrange multipliers.

The HQP solver has been widely used to generate motions on the HRP-2 robot in various contexts. A very typical example of inverse kinematics using both equalities and inequalities is summarized by Fig. 4.5. The considered tasks are (in hierarchical order) the joint limits ($\leq$), the COM inside the support polygon ($\leq$), the feet on the ground ($=$), the reaching ($=$), the FOV ($\leq$) and finally, a task keeping the COM on a small area in the center of the feet ($\leq$). The final configurations for various ball positions are given in Fig. 4.5. Other examples in inverse dynamics are discussed in Chapter 6.

The typical computation cost for the robot HRP-2 (36 DOF) is about $0.1ms$ when only equalities are considered and $2ms$ for a complete active search of 100 constraints.

4.3 Damping

The obtained HQP solver is very satisfying in term of performances and expressivity. However, the solver is also very accurate, which means that for highly ill-conditioned problems, it will doubtlessly answer with very high-value control. This is not acceptable in robotics (!), where we generally do not want to apply a $10^{15}rad/s$ velocity on the robot joints because the target to grasp is $1cm$ too far from the robot. In fact, a proper description of a hierarchical solver would be:

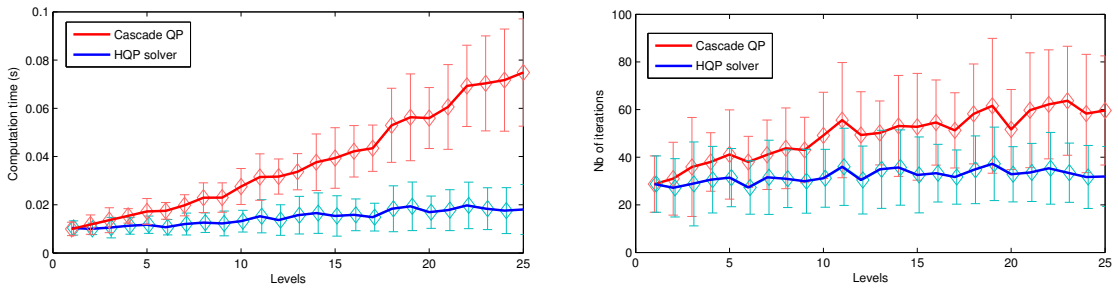
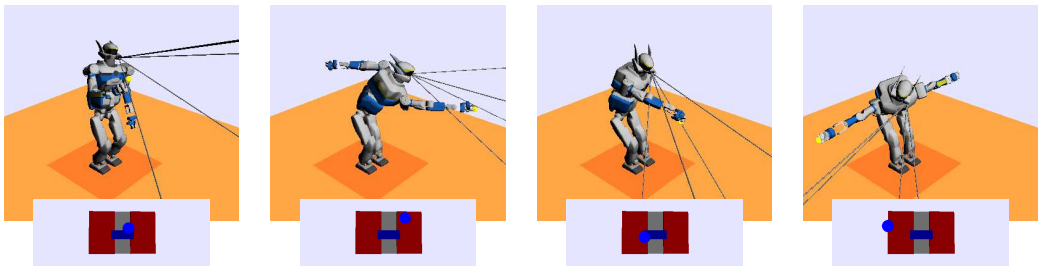


Figure 4.4 *– Comparison of the cost for solving inequality HQP using a cascade of QP or a HQP with hierarchical active search. The cost in milliseconds is plotted on the left, and in number of iteration in the active-search loop on the right. As in Fig. 4.3, a random set of HQP has been built with similar size but varying number of levels p. The cost increases with p when using cascade while it remains nearly constant using the HQP.*

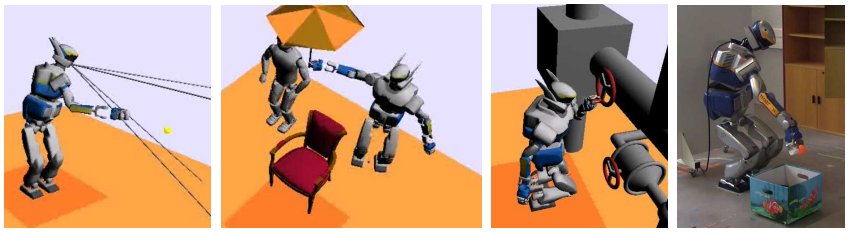


(t=2s) Ball in front (t=5s) Far left (t=7s) Front (t=10.4s) Far right

Figure 4.5 *– Snapshots of the robot motion and corresponding COM projection (blue point) in the support polygon. Each snapshot is captured at the end of a motion sequence. The FOV is displayed by the 4 lines passing by the center of the image projection. When the ball is in front, the robot reaches it while keeping it inside the FOV and keeping the COM in the central support area. On the far left, the FOV is kept but the COM has to leave the center of the support. On the far right, both the centering of the COM and the FOV are lost.*

Hierarchical quadratic programming

A. Escande, N. Mansard, P-B. Wieber



Submitted to International Journal of Robotics Research in Oct. 2012 [4]

Context:
This paper was written in close collaboration with A. Escande and P-B. Wieber. The original idea, proposed in the ANR JCJC R-Blink project, is to open the quadratic solver *black-box* and adapt the machinery to multi-objective programing. The journal submission follows an IEEE ICRA paper [28].

Motivations:
The pseudo-inverse, used in most of the task-function-based works, gives the solution of an implicit least-square quadratic program. When considering only one task, it is then straight-forward to take into account some inequality constraints, such as an obstacle or joint position and velocity limits. The extension to a hierarchy of tasks is not immediate and was proposed in [Kanoun 09b], by using a set of quadratic program in cascade. This formally defines the hierarchical quadratic problem and gives a naive resolution scheme that is very slow. This paper proposed to find an efficient resolution scheme to the cascade of quadratic problems.

Approach:
The study first focuses on equality-only quadratic program and proposes an efficient resolution method based on a dedicated extension of the matrix complete decomposition classically used to compute the pseudo inverse. The method is five times faster than the classical concurrent solution to solve a hierarchical quadratic program.
An active search dedicated to the hierarchical structure is then proposed to take into account a hierarchy of both equality and inequality. The solution avoids the problem of multiple activation and deactivation generally encountered with cascade of solvers. It is also much faster and straightforward to adapt to real-time constraints.
Incidental results are also given to prove the complexity, the continuity and the stability.

Results and contributions:
The dedicated solver is five times faster than the classical methods when considering only equalities. The inequality active search implies three times more computations than the equality solver, which keeps a computation cost admissible for closed-loop control. Experimentally, we also shown that considering a hierarchy simplifies the equality problem resolution and implies a very light extra cost when considering inequalities. Several example of use with the HRP-2 models are also given.

Limitations and perspectives:
It is difficult to damp the obtained solver: the classical Tikhonov regularization does not seems to be compatible with the active-search loop.

- apply at best the top priority objectives
- except if they are *stupid*, in that case do not consider them.

Of course, this notion of *stupid* has to be written as mathematical property to be handle by the solver. A classical solution for one task is to consider *stupid* to be equivalent to very high values of the optimum. In that case, a damping term is added to penalize high values. Problem (4.7) becomes:

$$\min_x ||A_1 x - b_1||^2 + \eta ||x||^2 \tag{4.17}$$

However, we are not able yet to propose an extension of this regularization to a inequality HQP. In particular, we did not find yet a good solution to take into account the regularization in the active-search loop. The damping can always be applied at the least-priority level. A practical solution is then to put all the tasks that might come to singularity in the least-priority level, and damp them to prevent any numerical problem during the inversion.

The problem of damping is closely related to the problem of continuously sequencing two set of tasks, and will be discussed in the next chapter.

4.4 Conclusion

In this chapter, we have proposed two solutions to handle inequalities inside a hierarchy of tasks. The first solution can only consider configuration-based inequalities (joint limits, obstacle but not actuator limits) and is computationally very expensive. However, it produces a very smooth behavior and is very easy to implement. It could be an interesting solution for simple systems or for smoothing a sequence of tasks (see next chapter). The second solution is based on active-search algorithms. It is very efficient (1kHz whole-body control is possible with HRP-2, which has never been done anywhere before). However, we did not find yet a good solution to use the classical damping term in this context. Therefore, it is sensible to the proximity of singularity and might not be safe to apply to control the physical robot. In the particular case where the singular task is at the least-priority level, a proper regularization is still possible.

No complete method is available yet, but the second approach is very close to come to a complete method and has already been applied in various contexts (see Chapter 12 for example). Many exciting perspectives arise from the results obtained here. Perspectives coming from direct applications are developed in the next two chapters. In particular, we had only the opportunity to validate the solver on humanoid robots and avatar, while it seems very suitable for example for team of fixed manipulators cooperating to assemble an object. Concerning the hierarchical problem formulation and resolution, the extension to more complex classes of numerical problems is very appealing. In particular, the extension to conic problems is very interesting for robotics modeling [Boyd 07, Wensing 13].

Inverse kinematics (and similarly, inverse dynamics) implies a linear approximation of the system dynamics, which is introduced to reduce the computation cost and to reach computation rate needed to control in real-time a complex robot. The instantaneous linearization also hides an important complexity carried by the future robot trajectory. However, in many setups, it is not possible to hide this complexity and the optimal control problem has to be considered at some point. In particular, to stabilize the humanoid balance, the conditions have to be written over the future trajectory of the system and does not survive to the linearization.

For walking, the robot typically follow a trajectory in the COM space, which is computed by solving an optimal control problem on a reduced-dynamics system, using model-predictive control (MPC). The two problems (inverse kinematics handling the spatial dimension of the system and walking pattern generation handling the temporal dimension of the system) are yet artificially decoupled for computational reasons. The insertion of the balance MPC within the whole body solver will be mandatory in the future.

Chapter 5

Continuity

During the execution of a given set of tasks, the inverse-based control schemes presented in the previous chapter are continuous with respect to time. However, the continuity is not ensured when the set of tasks is changed, when a task is added, removed or when the priority between two tasks is swapped. This chapter proposes some solutions to enforce the continuity of the control law at the transitions of a task sequence. We first give the foundations of the problem in Section 5.1 and give a first complete theoretical solution in Section 5.2. However, this solution is not compatible with the damping and cannot be applied in practice. Two practical solutions are then proposed in Section 5.3.

5.1 Bases

The control laws presented before, (3.26) for example, are discretized during the integration by the robot, the simulator or the motion planner. The continuity of the discrete sequence of $\dot{q}$ is not a relevant property. Two properties can be used to properly characterize the control. The continuity of the support function $t \to \dot{q}(t)$ can be discussed; or the rate of change between two samples of the sequence can be considered.

Continuous support function

The continuity of the support function does not ensure by itself any property on the rate of change between two samples. The Lipschitz continuity should rather be considered. However, when considering the continuity at a finite set of transition points, the frequency of the inherent task variations is generally considered negligible compared to the transition frequency. Therefore, if the function used to smooth the transition is *sufficiently* continuous, it is generally considered that the resulting properties on the discrete rate of change are satisfying. Moreover, a small discrete rate of change does not ensure a good behavior. When the continuity of the supporting function is not ensured, I often empirically notice that even a small rate of change can lead to oscillations around the transition border (see *e.g.* Fig. 4.1).

Bounded rate of change

The rate of change can be limited by adding an inequality to the QP, typically:

$$\min_{\dot{q}} ||J\dot{q} - \dot{e}^*|| \tag{5.1}$$

$$\text{subject to} \quad \delta t \ \underline{\ddot{q}} \leq \dot{q} - \dot{q}_{t-\delta t} \leq \delta t \ \overline{\ddot{q}}$$

where $\dot{q}_{t-\delta t}$ is the measured velocity at the previous cycle, δt is the sampling interval and $\underline{\ddot{q}}, \overline{\ddot{q}}$ are the bounds on the rate of change.

The bounds are imposed on each component of $\dot{q}$. The constraint is thus a small box (scale factor δt) around the previous control. If the unconstrained optimum suddenly jumps due to a transition, the small box will slowly move from the control before the transition to the control after the transition. By tuning the size of the box, the rate of change and the duration of the transition is controlled.

The constraint is a bound on the QP variables. A similar bound is imposed to enforce the joint limits, but with a $\frac{1}{\delta t}$ factor rather than a δt factor. A bound on the velocity (factor 1) can also be added to enforce the maximal velocity. Together with the bounds on the position and on the acceleration, it can also be seen as a dual way to damp the system [Maes 10].

A problem can be expected during the resolution: due to the size of the *box* in which the optimum is constrained, the solution would likely occurs at one corner of the box. The active search would then have to search for the proper corner of the box. Poor performances are expected since the active search is not well fitted for such a discrete search. Moreover, small changes of the unconstrained optimum would make the solution jump from one corner to the other, increasing the cost on several iterations. Moreover, the numerical behavior of the algorithm would also be problematic if such jumps frequently occur.

Instead of a box, the solution can be constrained into a ball:

$$||\dot{q} - \dot{q}_{t-\delta t}|| \leq \delta t \ \overline{\ddot{q}} \tag{5.2}$$

The constraint is not linear anymore but quadratic. It can be solved by a quadratic-constrained quadratic-problem solver [Anjos 12]. It would typically be written under a conic shape and solved by a conic-problem solver. The extension of the work presented in this thesis to conic formulation is one of the interesting technical perspectives. Conic problems are a simple generalization of linear problems. A very informative constructive formulation is given in [Nemirovski 94]. See [Boyd 07, Wensing 13] for application in robotics. However, if the problem structures are similar, active-search algorithm cannot be applied to conic problems. The solvers rather use interior-point methods. Extension to hierarchical problems should be feasible, using the HQP as the internal linear solver of the interior-point algorithm. However, it is still a long development to do.

In this chapter, we only consider the continuity of the support function and use smooth slowly-varying transition functions. Bounding the rate of change is definitely an appealing solution, for which we already see some problems but that deserves a try. This will be let as a perspective.

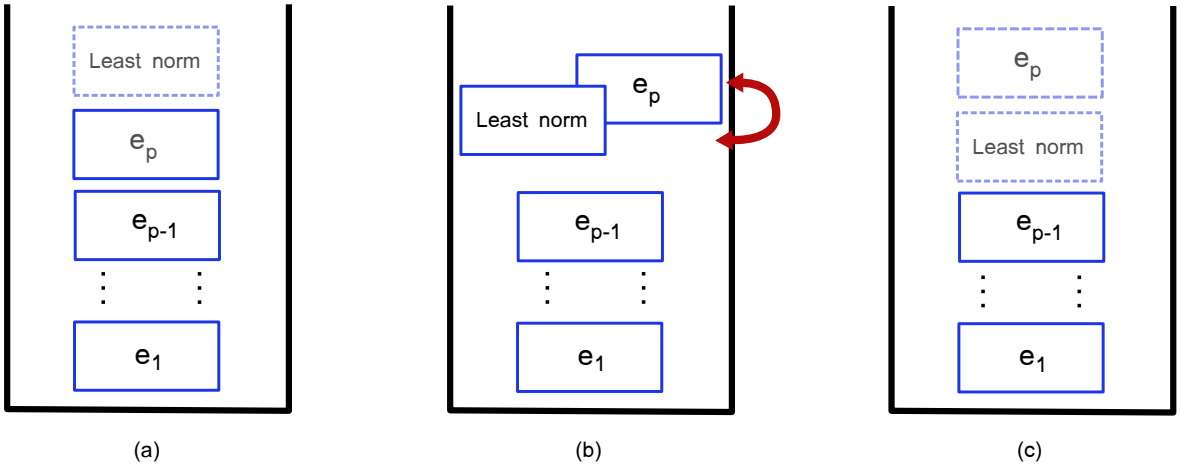


Figure 5.1 *– Removal as a swap operation. (a) Any hierarchy e_1 .. e_p is implicitly accounting for an implicit least-norm task after the last level. (b) The least-priority level e_p is swapped with the implicit least-norm task. This is equivalent to remove e_p. (c) The task e_p with a priority lower than the least-norm task has no influence on the optimum. The task e_p is actually removed from the hierarchy and the stack (c) is equivalent to the stack containing the $p-1$ first tasks.*

5.2 Theory

Reference paper [22] (see also [8, 14, 34, 45, 61])

Three operations can lead to discontinuities: introducing a task, removing a task and swapping the priority between two tasks. Examples of discontinuities in each case are given in [8].

Discontinuity intuitions: The discontinuity at the insertion of a task $(e, J, \dot{e}^*)$ is the easiest one to explain: the velocity in the task space e before the insertion is $J\dot{q}$ and after the insertion is $\dot{e}^*$. It is also the easiest one to smooth [45]: a transition is added in the task space between the velocity before the insertion and the velocity after the insertion, using for example a second-order differential equation on $\ddot{e}^*$ [Soueres 03]. If the task is inserted with a higher priority than an incompatible task, a discontinuity also appears in the conflicting spaces.

Similarly, the discontinuity at the removal is easy to understand. It is more difficult to smooth since the velocity after the removal is not computed and is varying during the transition due to the non linearities (variations of J) and the evolutions of the other tasks.

The discontinuities of the swap are due to the reallocation of the conflicting DOF from the task having priority before the swap to the task having priority after the swap.

Swap as the common factor: Rather than trying to find a solution to smooth each cases, it was proposed in [22] to take the three operations back to the swap. Indeed, among the

possible optima of a given HQP, the least-norm one is always chosen[1]. This is equivalent to consider that at the least-priority level, a task tries to bring the velocity to zero. This implicit last level is called the *least-norm* task. Consider a hierarchy of p tasks (the *least-norm* task is at level $p+1$). Swapping task p with the *least-norm* task is equivalent to remove it (see Fig. 5.1). Inserting a new task e_{p+1} after the *least-norm* task and then swapping the priority between these two tasks is equivalent to inserting e_{p+1}. Then inserting or removing a task at any level of priority can be achieved by several swaps.

Therefore, if the swap operation can be achieved in a continuous manner, the continuous insertion and removal can be achieved by combining smooth swaps. We can now consider a reduced problem where the hierarchy is composed of only two tasks to be swapped.

Smooth swaps: The hierarchy is known to be obtained when the weight ratio of a weighted sum reaches the limit. Consider the least-square optimum of the following weight-sum QP:

$$\dot{q}^*_\rho(\rho_A, \rho_B) = \min_{\dot{q}} \rho_a ||J_A\dot{q} - \dot{e}^*_A||^2 + \rho_b ||J_B\dot{q} - \dot{e}^*_B||^2 \tag{5.3}$$

The hierarchy $e_A \prec e_B$ (denoted $\dot{q}^*_{A|B}$) is obtained when ρ_A is infinitely higher than ρ_B and reciprocally:

$$\dot{q}^*_{A|B} = \lim_{\rho_B/\rho_A \to 0} \dot{q}^*_\rho(\rho_A, \rho_B) \tag{5.4}$$

$$\dot{q}^*_{B|A} = \lim_{\rho_B/\rho_A \to +\infty} \dot{q}^*_\rho(\rho_A, \rho_B) \tag{5.5}$$

The continuous swap can then be achieved by continuously varying the weight ratio. For example, the function $h \to \dot{q}^*_\rho(-log(h), -log(1-h))$ continuously varies from $\dot{q}_{A|B}$ when h tends toward 0 to $\dot{q}_{B|A}$ when h tends toward 1. The transition is then performed by moving h from 0 to 1.

When both tasks are compatible, $\dot{q}^*_{A|B} = \dot{q}^*_{B|A}$, the weights have no influence and the transition can be instantaneous without discontinuity.

5.3 Practice

Reference papers [8, 22] (see also [25, 34, 12])

5.3.1 Damping

In practice, a damping factor is always added to regularize the QP close to singular regions. The QP during the transition is then written:

$$\min_{\dot{q}} \rho_a ||J_A\dot{q} - \dot{e}^*_A||^2 + \rho_b ||J_B\dot{q} - \dot{e}^*_B||^2 + \eta||\dot{q}||^2 \tag{5.6}$$

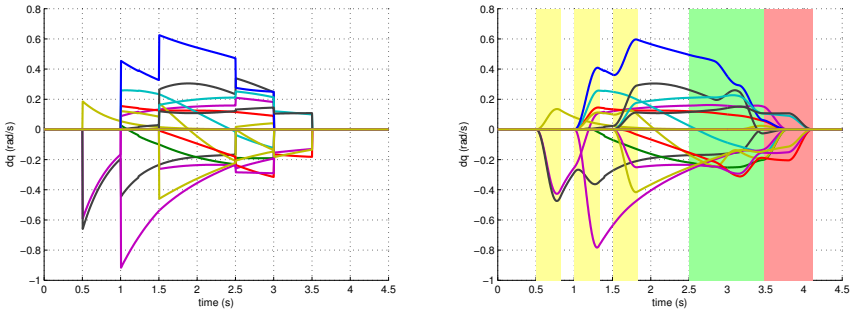
where η is the damping parameter (typically $\eta = 10^{-2}$).

The optimum of this QP does not tends toward the optimum of a hierarchy containing e_A and e_B. Indeed, when ρ_A is much smaller than η, the term A is numerically removed from

[1] This is due to the use of the pseudo-inverse, also called *least-square* inverse and clearly appears when considering the HCOD in (4.14)

Analysis of the discontinuities in prioritized tasks-space control under discreet task scheduling operations

F. Keith, P-B. Wieber, N. Mansard, A. Kheddar



IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS'11) [22]

Context:
This paper was written during the PhD of F. Keith in collaboration with P-B. Wieber.

Motivations:
Smoothing the control law at the transitions of a task sequence is a classical problem. However, few strong solutions have been proposed because, in practice, the physical motors of the robot are acting as a low-pass filter that is doing most of the smoothing. The limitation of the previous approaches was revealed while using a numerical solver to optimize the parameters of a task sequence. Discontinuities in the control law correspond to irregular points that reduces the convergence rate of the solver. The following method was proposed as a generic solution to the task-sequence smoothing problem, and more specifically validated in the context of optimizing a sequence of tasks.

Approach:
The problem of insertion and removal of one element from a stack of tasks is first reduced to the problem of swapping the priorities. Insertion is then equivalent to swapping the new task with the least-square objectives that implicitly occupy the lowest priority of the stack. The continuity can then be studied on a two-levels stack.
The hierarchy between the two levels is known to be equivalent to the limit of a weighted sum of the two tasks where the ratio between the weights are going to zero or to infinity. Inverting the priority is then obtained by smoothly inverting the weight ratio.

Results and contributions:
The method is a clean formulation of the task-sequence smoothing problem. It clearly defines the process in terms of optimization objectives and gives properties on the behavior during the transition.

Limitations and perspectives:
The method is not compatible with the classical damping solutions used to regularize a hierarchical problem close to singular regions. In that case, another solution is proposed, but it is not properly formalized as an optimization problem and cannot ensure any theoretical property (apart from the continuity) during the transition. Moreover, it then would not be possible to adapt it to a proper HQP solver. Hierarchical programming with damping and continuity at transition is still an open topic.

the sum and e_A is not taken into account any more. Experimental evidences of this effect are given in [22].

Once more, we have a solution that is properly written and ensures all the required properties, but that we cannot regularize in the neighborhood of singularities.

In fact, the damping process is very closely related to the continuity. First, removing a task by swapping it with the implicit *least-norm* task is equivalent to increasing the damping factor of this task to the infinity. On the dual hand, the damping can be seen as a way to smooth the transition through the singularity border.

Moreover, we have considered with P-B. Wieber the possibility to damp a task not by the classical $\eta||\dot{q}||^2$ but using the next task through a term $\eta||J_2\dot{q} - e_2||^2$. This takes the hierarchy back to a weighted sum without the numerical problems due to large weight ratios. If this solution does solve totally the problem of automatic regularization (in particular, two decoupled tasks cannot be used to regularize each other), it would correspond exactly to the formalization of the continuity problem presented above.

In conclusion, we have a solution to enforce the continuity at task transitions if and only if we have a solution to ensure the continuity when coming across a singularity. This solution is not completely functional yet. Two practical alternatives have then be proposed.

5.3.2 First practical method

In [22], it was proposed to enforce the transition during a swap by using an explicit homotopy between the optimum computed for the two hierarchical orders:

$$\dot{q}(h) = f(h)\dot{q}^*_{A|B} + f(1-h)\dot{q}^*_{B|A} \tag{5.7}$$

with h the transition parameter varying from 0 to 1 and $f : h \to f(h)$ any continuous strictly monotonic function from $f(0) = 0$ to $f(1) = 1$ (*e.g.* $x \to x$).

The swap (5.7) is for a two-tasks hierarchy. The insertion of a task at a given priority level is realized by successive swaps: the task goes down to its rank (following the bubble-sort principle). Two tasks can be successively inserted but their swaps cannot overlap: a delay (of the duration of one swap) after the beginning of an insertion is enforced. For the same reason, a task cannot be removed while a task is inserted to avoid that two *bubbles* cross each other.

The continuity is enforced with a very low computation overhead. Experimental results can be found in [22].

5.3.3 Second practical method

We simultaneously worked with J. Park to apply the generic solution proposed in [12] and described in Chapter 4 to the specific continuity problem. The idea is to perform an homotopy between the stack with and without the task to be inserted, or between the two orders before and after the swap. Of course, the homotopy can be extended to handle several simultaneous operations.

A comparison between the two methods is summarized in Fig. 5.2. During simultaneous operations, the first method composes several local homotopies while the second method uses one large homotopy handling all the cases. The second method is thus more expensive.

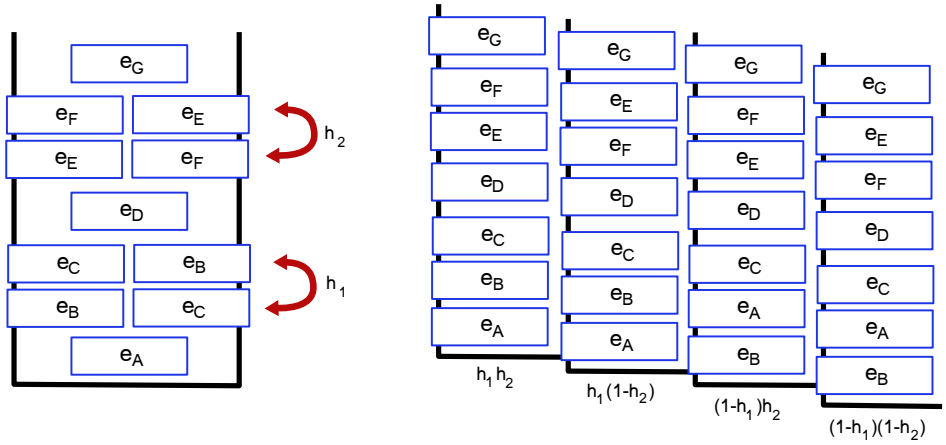


Figure 5.2 *– Comparison between the two practical methods for two simultaneous swaps. (left) First method: the homotopy is performed between each order of the two levels B/C and E/F. (right) Second method: the homotopy is performed between every possible orders of B/C/E/F. For several simultaneous swaps, the first method is more efficient, but the second method can be used to directly introduce a task at any priority level.*

However, it is able to handle directly insertion at and removal from any level, and does not introduce any delay between two operations. Moreover, an approximation called *one-path* approach was proposed to lower the cost if too many operations occur at the same time.

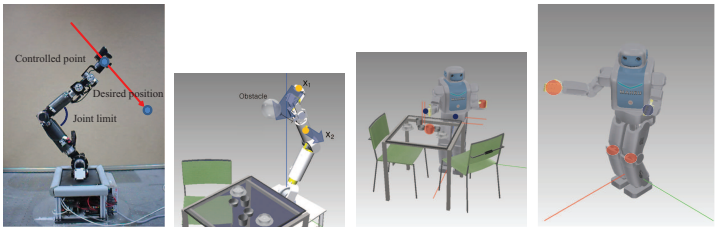
5.4 Conclusion

The continuity problem is well defined using a continuous variation of weight to a ratio limit. However, due to the lack of a *good* solution to automatically regularize the numerical problem close to singular regions, we are not able to propose a solution that handles the continuity during task transitions and provides a good robustness to singularities. We have some elements to believe that the proposed solution will be sufficient if a proper damping is established. In the meantime, two practical methods have been proposed that can be applied to actually smooth any transitions.

A complete solution to the continuity problem finally relies on the problem of finding a proper automatic regularization of the numerical problems encountered in robotics, the so-called *damping* problem described in the previous chapter.

Intermediate desired value approach for task transition of robots in kinematic control

J. Lee, J. Park, N. Mansard



IEEE Transaction on Robotics, 2012 [8]

Context:
The paper comes from the work about the continuity realized at Stanford [12]. The initial formulation immediately interested J. Park (post-doc in O. Khatib's team at that time) for smoothing the control law at the beginning of a task. The work was then completed during the Master Thesis of J. Lee, in Korea.

Motivations:
The smoothing of varying-dimension tasks, proposed first for activation and deactivation of inequality constraints, can be directly adapted for activation and deactivation during a task sequence. The trigger is then parametrized by the time instead of the robot configuration.

Approach:
The paper first proposes to use directly the results of [12]. To reduce the computation cost, an approximation called *"one-path approach"* is then proposed. A strong experimental study in various contexts concludes the paper.

Results and contributions:
The paper gives a complete solution of the problem of the continuity at task insertion. The solution is easy to apply. The complexity in extreme cases is reduced by the approximation.

Limitations and perspectives:
The method is not compatible with the work about quadratic programming [4] and cannot comprehend inequality constraints.

Chapter 6

Dynamics

The motion-generation formulations proposed in the previous chapters only consider the geometry of the robot. The dynamics is now added. For humanoid robots, this is critical, since the robot balance is depending on its dynamics. Inverse dynamics increases a lot the size of the numerical problems to consider. A first part of our work, discussed in Section 6.1 focuses on the formulation of the problem, in order to take into account all the modalities of the humanoid robots while keeping a reasonable computation cost. In particular, a reduction of the hierarchical problem is proposed in Section 6.2 to enable 200Hz computations. The main objective is to apply it as a closed-loop control scheme on the real robot. This is still an open work, as it will be discussed in Section 6.3.

6.1 Problem formulation

Reference papers [6, 24] (see also [18, 19, 21, 47])

6.1.1 Simple manipulator

For a fully-actuated manipulator robot, the dynamics of the system can be written:

$$A(q)\ddot{q} + b(q, \dot{q}) = \tau \tag{6.1}$$

where A is the generalized inertia matrix of the system, b is the dynamic drift composed of Coriolis forces and gravity and τ is the vector of the joint torques that have to be produced by the motors. A is positive definite thus invertible. The equation can also be written the other way around *i.e.* computing $\ddot{q}$ with respect to τ. This way is called the inverse dynamics. τ is generally considered as the control input to be provided to the motor for low-level regulation. In that case, $\ddot{q}$ is a reference acceleration. It can be computed by (second order) inverse kinematics, starting from the derivative of the direct kinematics:

$$J\ddot{q} = \ddot{e} - \dot{J}\dot{q} \tag{6.2}$$

This equation has the same structure as (3.15): $v = Gu + \mu$ with v the task evolution, u the system input, G the differential map between v and u and μ an intrinsic drift that has to be estimated and compensated. The inversion has the same shape as (3.16):

$$\ddot{q} = J^{+}(\ddot{e}^{*} - \dot{J}\dot{q}) \tag{6.3}$$

from which τ can be deduced.

Both $\ddot{q}$ and τ are variables of the system that are coupled by (6.1). They are computed above by a cascade of two steps that are decoupled thanks to A being invertible. However, if some constraints are imposed on τ (actuation limits or underactuation), they will also indirectly constrain $\ddot{q}$. It is thus more interesting to consider the inversion of both the kinematics and dynamics in a same problem:

$$\min_{\ddot{q},\tau} ||J\ddot{q} - \ddot{e}^{*} + \dot{J}\dot{q}|| \tag{6.4}$$

$$\text{subject to} \quad A\ddot{q} + b = \tau$$

When J is not full-column rank, redundancy on $\ddot{q}$ and τ can be exploited. Rather than imposing a least norm on one or both variables, it is often proposed to comply with Gauss's principle that states that the system is the closest to the evolution of the unconstrained system, *i.e.* that

$$\ddot{q} = -A^{-1}b \tag{6.5}$$

[Bruyninckx 00]. This is equivalent to impose a least pseudo energy $\ddot{q}^{T}A\ddot{q} = \tau^{T}A^{-1}\tau$ [Park 06a]. When no additional constraints are considered, the solution of the system is obtained by:

$$\tau^{*} = (JA^{-1})^{\#_A}(\ddot{e}^{*} - \dot{J}\dot{q}) + b = J^{T}f^{*} + b \tag{6.6}$$

$$\ddot{q}^{*} = J^{\#_{A^{-1}}}(\ddot{e}^{*} - \dot{J}\dot{q}) \tag{6.7}$$

with $f^{*} = \Lambda(\ddot{e}^{*} - \dot{J}\dot{q})$ and $\Lambda = (JA^{-1}J^{T})^{+}$ is the equivalent inertia in the task space. The first equation is interesting because of its similitude with the action of a force f acting on the robot body: both f and the equivalent inertia times acceleration $\Lambda\ddot{x}$ produce the same torques, for the task function being the application point position $e = x$. This is called dynamical consistency [Khatib 87]. The second equality is interesting because it is evidently the optimum of (6.4), with a particular choice of the weighting of J. Both optima are coupled since $AJ^{\#_{A^{-1}}} = (JA^{-1})^{\#_A}$.

6.1.2 Humanoid robot

The dynamics of the humanoid robot generally considers two additional terms. First, the robot is underactuated. Rather than imposing a part of the torque vector to be zero, the torques are often reduced to $S^{T}\tau$ (where $S = [0\ I]$ is the selection matrix of the actuated DOF). Second, the robot is in contact with the environment. The contact forces have also to be accounted as variables of the system, like the torques and the acceleration. Indeed, due to Newton reaction law, the robot decides which are the environment forces at the same time it decides what are the joint torques. Of course, the relation depends on the interaction model (whether the robot is contacting a rigid surface or a damper). Force sensors measure

the application of the chosen forces at the previous control cycle and can be used to update the interaction model [Park 06b].

The dynamics of the humanoid robot subject to rigid contacts is finally:

$$\begin{aligned} A(q)\ddot{q} + b(q,\dot{q}) &= S^T\tau + J_c^T f_c & (6.8)\\ J_c\ddot{q} + \dot{J}_c\dot{q} &= 0 & (6.9)\end{aligned}$$

with

$$f_c \in \mathcal{C}_f \quad (6.10)$$

The forces f_c are applied by the environment on the robot contact points x_c, whose Jacobians with respect to the robot configuration are denoted by J_c. The forces are constrained to be inside the contact cones $\mathcal{C}_f$ while the contact points are not moving due to the rigid contact hypothesis. When several contact points arise, the J_c and f_c are stacked and keep the same shape in the equation.

Due to the cones, the equations are not linear but conic. The cones can be linearized by considering a polyhedron approximation with a finite set of facets [Bouyarmane 11b]; or more roughly, the cone can be approximated to the half space:

$$f_c^{\perp} \geq 0 \quad (6.11)$$

This second hypothesis is used below for two reasons. First, HRP-2 is experimentally rarely sliding. We only consider enough friction to prevent any slides in open loop. Second, the linearized cones are very expensive to consider, while this cost is very artificial. Indeed, the original conic constraint is very simple, cheap to check and to enforce inside a conic solver. We prefer yet to keep a low computation cost with half-space approximation, and to use a real conic solver if the cones become an issue.

6.1.3 Problem resolution

We finally consider the resolution of a set of tasks $e_1, \ldots, e_p$ and of $c_1 \ldots c_k$ contact points. The HQP problem is written:

$$\underset{\ddot{q},\tau,f_c}{lsprox}\ (6.12\#\text{dyn}) \prec (6.12\#\text{f}) \prec (6.12\#\text{c}) \prec (6.12\#e_1) \prec \cdots \prec (6.12\#e_p) \prec (6.12\#\text{v}) \quad (6.12)$$

$$\begin{aligned} \text{subject to} \quad \ddot{q} &= -K_v\dot{q} & (6.12\#\text{v})\\ J_p\ddot{q} &= \ddot{e}_p^* - \dot{J}_p\dot{q} & (6.12\#e_p)\\ &\vdots & \\ J_1\ddot{q} &= \ddot{e}_1^* - \dot{J}_1\dot{q} & (6.12\#e_1)\\ J_c\ddot{q} &= -\dot{J}_c\dot{q} & (6.12\#\text{c})\\ f_c^{\perp} &\geq 0 & (6.12\#\text{f})\\ A\ddot{q} + b &= S^T\tau + \sum_{c=1}^{k} J_c f_c & (6.12\#\text{dyn})\end{aligned}$$

Only τ is needed for the control. However, this explicit formulation computes the value of $\ddot{q}$ and f_c as well. The last level is a friction term. It is required by the task-function approach to ensure that the control law is stable [Samson 91]. When the system is intrinsically stable (for example, when intrinsic joint friction stabilizes the control), this last level can be replaced by the Gauss's criteria (6.5).

In [6], we have reduced the number of force variables by considering the sum of all forces and torques on each body in contact rather than the contact forces individually. The criteria (6.12#f) can also be brought back to the classical (zero-moment point) ZMP constraint when all the contact points are on a same horizontal plane.

6.2 Condensation

Reference paper [17]

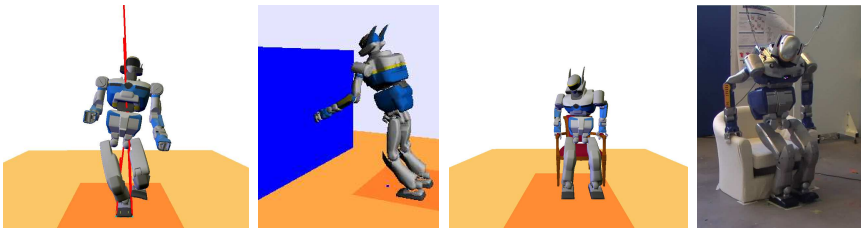
The system (6.12) is very large: for HRP-2 with two feet on the ground, it is typically 90 variables, 90 equality constraints, 10 inequality and 20 to 40 task constraints. Moreover, the constraints (6.12#c) and (6.12#f) due to the dynamics might be ill conditioned. A typical example of the *chimney climber* is given on Fig. 6.1. Such a conditioning problem is typical of robotics: they represent a real physical effect that is impossible to achieve with the characteristics of our systems. They can be simply isolated by cutting the too small singular values. However, it is not so easy to do inside the HQP solver, that imposes its own threshold.

To both reduce the size of the system and avoid the side effects of the small singular values, the HQP (6.12) is condensed beforehand in a dynamically-consistent manner. Basically, the idea is to change the variables $\ddot{q}, \tau, f_c$ into two new decoupled variables u the free motion and ϕ the free actuation. The motion u is free of any constraint while the actuation ϕ is only subject to the bound limitations (positive forces and limited torques). If no joint torque limitations have to be considered (the robot is *a priori* powerful enough for a given motion), this constraint and the corresponding variables could be reduced once more by projecting all the cones (or half spaces) at a central point [Bretl 08]. The details of the condensation are given in [17].

A typical example of use is the motion *standing lotus*, performed from motion capture data. The considered tasks are the position of the hands, the orientation of the head, the posture and, during single support phases, the position and orientation of the flying foot. The measured computation times are summarized on Fig. 6.2. Basically, the time spent in the solver is reduced from $25ms$ to $5ms$ by using the condensation. The finally obtained computation times are smaller than the classical methods for example using the formulation given in [Sentis 07]. In total, the cost of the inverse dynamics for HRP-2 is around $5ms$ thus nearly fast enough for application in real time on the robot. This is much faster than our implementation of [Sentis 07]. In [de Lasa 10], times below $1ms$ were achieved on a very strong computer (typically used in graphical animation but not possible to embed on a humanoid with a realistic setup). We did not yet experimentally compare the two schemes. A first comparison was performed in inverse kinematics (see Chapter 4 and [4] for details) that gives close computation times with a small advantage for our approach and a much better numeric behavior. The condensation should increase the advantage but should be measured.

Dynamic whole-body motion generation
under rigid contacts and other unilateral constraints

L. Saab, O. Ramos, F. Keith, N.Mansard, P. Souères, J-Y. Fourquet



IEEE Transactions on Robotics [6]

Context:
This is one of the two papers written during L. Saab thesis, who worked on the generation of motion of anthropomorphic systems for ergonomics and virtual prototyping.

Motivations:
Task-based motion generation enables fast development of whole-body movements. When accounting only for the geometry of the system, the movements are very rough. This is especially true when generating human-plausible motions. The inverse-dynamics solver was developed to generate dynamically-consistent movement from a hierarchy of tasks.

Approach:
The motion generator is based on the HQP solver [4]. The variables are the contact forces and joint torques and accelerations. These variables must fulfill the dynamics equation and satisfy the given contact model (only rigid contact was tested yet). When all the contact points are included in a horizontal plane, the contact constraint is equivalent to the classical ZMP constraint. The tasks are finally introduced in the solver by imposing the acceleration in the operational space to be equal to a specified value.

Results and contributions:
The solver was used to generate fast motions with HRP-2, large upper-body motion during the walk and multi-contact motion (like using the armrests to sit in a chair). It was also broadly used to generate the *Novela* dance show from motion-capture data [2]. The motions produced with this solver are much easier to apply on the physical robot (since they comply with the true dynamics). Some preliminary results were also obtained for human-plausible movements [7].

Limitations and perspectives:
The major perspective is the real-time application to control the physical robot. The dynamics solver is slow and subject to numerical ill conditioning. These problems should be addressed before applying it in closed loop on the real robot and a first proposition was done in [17], which in turn revealed some other perspectives.
Taking into account the dynamics means that the solver also takes into account the intrinsic instability of the biped robot. The ZMP-like constraint is not sufficient to avoid the unstable mode since it does not apprehend the future. Preview control must be linked with the whole-body computations to handle it.

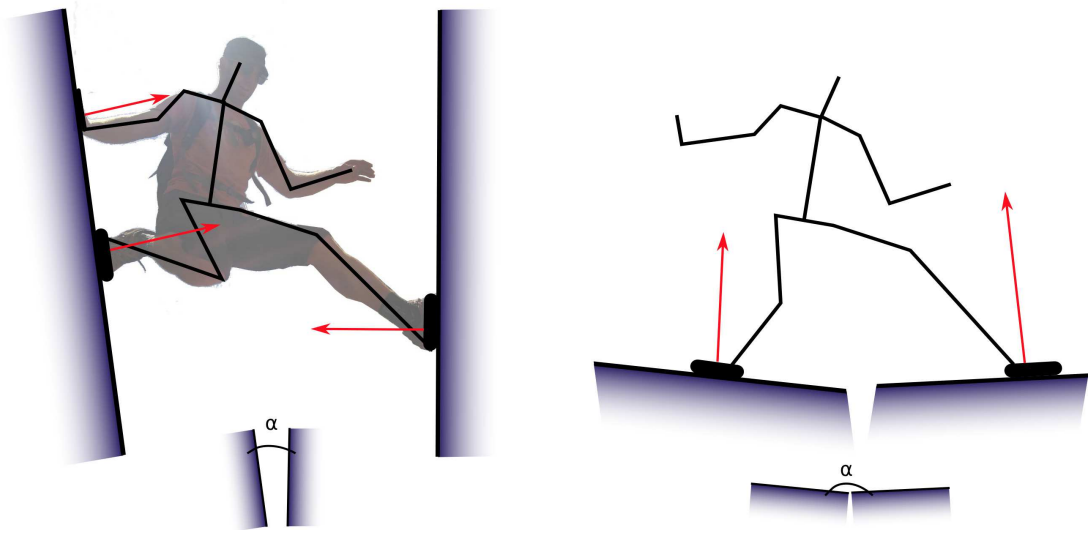


Figure 6.1 *– The chimney climber ill-conditioning: when both feet have a very small respective orientation angle, a force opposition effect appears that can be strongly used to maintain the balance. The same numerical effect exists with very large angles, but is not relevant on a mechanical point of view. However, the solver will equivalently use this effect in both cases, leading to very high (typically* $10^8 N$*) internal forces.*

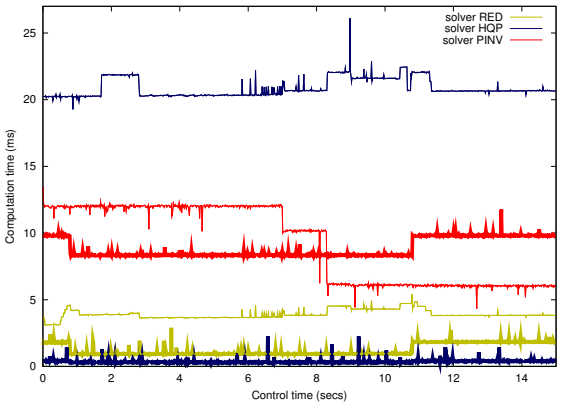
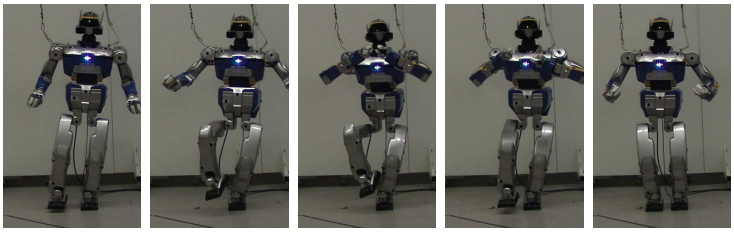


Figure 6.2 *– Experiment B: formulation time (bold) and solver time (non-bold). Three solvers are compared. [RED] the reduced HQP-based inverse dynamics proposed in [17]; [HQP] the extensive HQP-based inverse dynamics proposed in [6]; [PINV] a more classical pseudo-inverse-based dynamics inverse similar to [Sentis 07] (that does not handle the unilateral force constraint. [HQP] is the more expensive with a very constant cost. [PINV] cost is shared between the formulation (contact projector computation) and the inversion itself. The repartition of the cost varies with the number of contacts. [RED] is the more efficient, with around 4ms per control cycle. The variation of the HQP resolution comes from the active search.*

A dedicated solver
for fast operational-space inverse dynamics

N. Mansard



IEEE Int. Conf. on Robotics and Automation (ICRA'12) [17]

Context:
This communication was written during the preparation of the HRP-2 dance, to solve a problem of numerical conditioning that was perturbing the active-search loop convergence. The resulting solver was intensively used by O. Ramos to realize the robot dance and by L. Saab for her PhD experiments.

Motivations:
In the possible values of the force, torque and acceleration variables, some create a motion, some create an internal force and the others are forbidden. The selection between these three possibilities is defined by a threshold on the singular values of the system. If the constraints are directly solved by the QP solver, the threshold is imposed by the active search and is very low, which leads to singular behavior. Simultaneously, the dimension of the complete problem is high with sparse matrices.

Approach:
The problem is condensed before sending it to the QP solver. Based on the work of O. Khatib, the condensation is done in a dynamically-consistent manner. The variables are reduced to the *motion* and *actuation* variables, which are decoupled and of minimal size. The dynamics and contact constraint matrices are reduced and dense. The task constraints are rewritten to depend on the *motion* variable instead of the acceleration variable.

Results and contributions:
The condensation removes the problem of conditioning in multiple-support phases. The computation cost is reduced by a factor of five, from 20ms to 4ms for the HRP-2 model.

Limitations and perspectives:
The condensation has been chosen to be dynamically consistent. It might be more efficient to search for a condensation that optimally uses the sparsity properties of the problem. In particular, the reduction of the torque and force variables might be improved.
The computation times are still too important to use the inverse-dynamics scheme in closed loop onboard the robot at the nominal frequency of 1kHz.

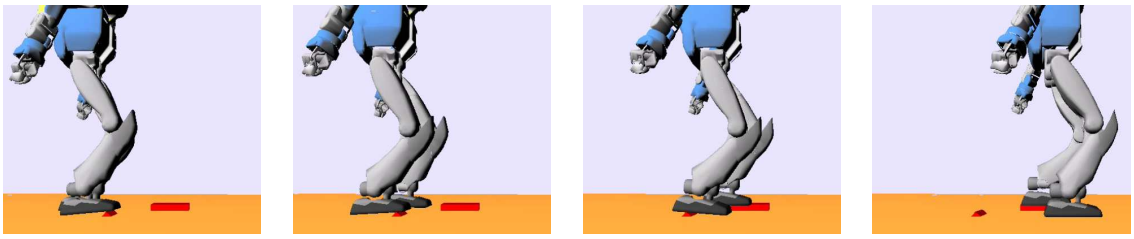


Figure 6.3 *– The inverse dynamics as a robust controller: Walking on a non-flat ground.*

6.3 Toward the robot

Reference paper [16]

The objective of the work described in this chapter is to build a method to control the robot dynamics. For that purpose, the computation cost is not the only aspect to solve. We should also demonstrate the interest of taking into account the whole dynamic model in the control and search for solutions to apply it on the robot.

In [16], we have proposed a first use of the inverse-dynamics scheme as a robust controller in simulation, to control the walk of HRP-2 on an unknown non-flat terrain. The hypotheses of the simulation are the following. The simulated robot is tracking the given reference torques (and accelerations). It also perfectly "perceives" the collision points that are used in the force part of the dynamics equation. The floor is a random distribution of small objects (2cm) on a horizontal plane (see Fig. 6.3). However, the model of the floor is not known from the controller, which uses the approximation that the floor is horizontal. The COM trajectory is computed at each control cycle by MPC using a classical inverse-pendulum linearization [Herdt 10]. The hierarchy of tasks (6.12) is composed of the COM task, one task to track the flying-foot placement (using the second order formulation of (3.14) as proposed in [de Lasa 10]) and one task to emphasize the robot posture and avoid drifts of the chest attitude. Contact points are added in (6.12) when a collision occurs between the flying foot and the ground during the foot landing. When three contact points are added, the foot landing ends because the foot movements are completely constrained. The contacts are relaxed when the walking pattern generator decides to take off.

The controller is then able to absorb the *unknown* rough terrain: the COM tracks the MPC trajectory even if the contact surfaces are not exactly those used by the inverse-pendulum model. The rough horizontal approximation is sufficient to ensure the proper regulation by the inverse-dynamics controller.

However, the simulation is only a first step toward the robot. HRP-2 (like most of other humanoids) is not able to track a reference joint-torque. Using the joint accelerations as control input is equivalent in the free space, but not in contact: in that second case, an acceleration-based control is not passive as it supposes a perfect answer of the contact model. In particular, for the walk proposed above, the contact points are needed. They could be measured using tactile sensors on the robot sole (a skin) [Mittendorfer 11, Takahashi 05] or partially reconstructed using several measurements of the force sensor in the robot ankle [Petrovskaya 07]. If these solutions are not enough, future generations of humanoid robots might bring some

more powerful tools. The DLR/Kuka LWR technology [AlbuSchaffer 07a] already applied to a biped robot [Ott 10] and a biped+torso robot offers a low-level joint-torque regulation by a 3kHz controller that might offer a sufficient bandwidth for the classical situations encountered with a humanoid. A major limit of this technology is its price. Moreover, despite their torque-sensing capabilities, these robots are still used with a high-gain low-level position controller. Alternatively, transparent actuators such as those waited in Aldebaran Romeo [Guizzo 10] might enable to reconstruct the joint torques from the motor current for a cheaper cost, and passively apply a reference torque with a larger bandwidth than the active LWR. On the other hand, adding springs into the actuation chain [Pratt 95, Tsagarakis 11, Grebenstein 11] enables to better absorb the impacts. However it raises other challenges on the control side [Sardellitti 12, Li 12]. The applied torques are reconstructed by measuring the deformation of the spring included in the actuators, which also enables to regulate the joint torques and to passively stabilize the robot at the impact. This is typically the approach used by Boston Dynamics [Raibert 08, Nelson 12]. In a sense, such an approach is already used in HRP-2 since spring-dampers are placed between the last motor and the sole to reduce the impact. This passivity is stabilized by an external control loop based on a simplification of the robot dynamics [Kajita 10a].

The inverse dynamics is a key tool for each of these three technologies, sensor-based torque control, transparent actuation or elastic actuation. In the three cases, the key is to add in the kinematic chain a machinery to measure the forces coming from the robot and environment interaction and to find the proper way to introduce these measurements in the inverse-dynamics control loop.

In conclusion, the work reported in this chapter enables to obtain the first real-time inverse-dynamics whole-body controller that now should be linked with the physical robot control loop. In particular, the connection with the sensors is the topic of the next chapter.

Chapter 7

Sensors

The methods presented in the three last chapters apply on a robot but also on avatars evolving in virtual worlds or in simulation to plan the robot motions. Most of the presented motions have been obtained with this last solution, to simplify the experimental process: the motion is generated off-line in simulation and is replicated by the physical robot using the joint encoders as the only loop closure.

In this chapter, we report the work realized to actually execute the motion on the physical robot in realistic conditions. This is mostly preliminary works: we are still trying to use the data extracted from the sensors in complex sensor-based control loops.

7.1 Force

Reference paper [33]

The force sensor is a relatively simple though expensive mechanism: a very rigid six-dimension spring is attached between two robot parts (typically the last motor of the robot kinematic chain and the effector), whose very small deformations due to applied forces are measured by strain gauge. The sensor is fast (1kHz on HRP-2), easy to unbiased and slightly noisy. It measures the three force components and three torque components applied to the spring, that is to say in the case of a sensor attached to the end effector, the sum of the forces applied by both the robot and the environment. If the forces applied by the robot are known (in inverse dynamics for example), the sensor can be used to evaluate the contact interaction model (elasticity of the contact surface [Park 06b] or force applied on the robot [Kaneko 12]). Or it can be used to regulate the force applied by the robot on a supposedly known environment.

Typically, a force sensor is used if moving in contact with a inverse-kinematics control scheme. An impedance filter [Hogan 84, Maeda 01] is used to transform the force information to a velocity reference. The motion of the end effector is modeled by a damped mass moving due to the forces recorded by the sensor:

(a) (b) (c) (d)

Figure 7.1 *– Four examples of force control. (a) Inverse dynamics: a force sensor is not necessary to move the robot in the null-space of the task (here a 3-DOF visual servo). (b) Impedance control to enforce a zero torque while moving the fridge door. (c) Impedance control to follow the human partner while walking. The footsteps are chosen to keep a reference position of the arm, leading the robot to dance with its human partner. (d) carrying a table while walking. The impedance control is once more used to follow the human operator.*

$$M\ddot{r} - B\dot{r} = \hat{\phi} - \phi^* \tag{7.1}$$

where r is the placement (six dimensions) of the mass, M its generalized inertia, B the damping factor, $\hat{\phi}$ is the force wrench (spatial force [Featherstone 08]) measured by the force sensor and ϕ^* is the reference force to apply. This differential equation is integrated once (*e.g.* using an Euler integrator) to obtain the reference velocity $\dot{r}^*$ tracked by the inverse-kinematics control scheme.

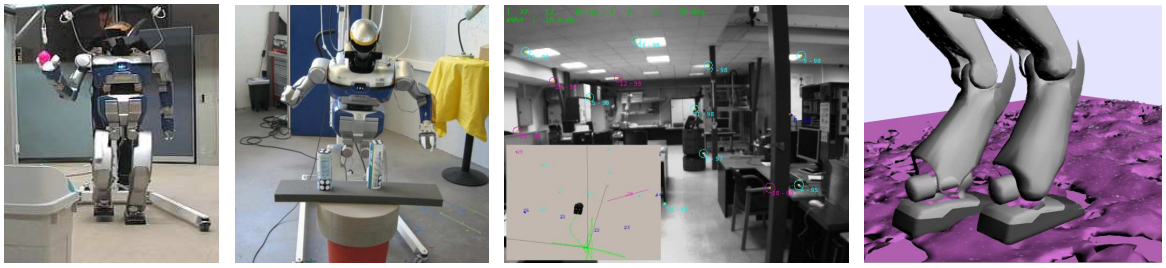
In [33], the inertia and damping matrices of the impedance filters are automatically adjusted to correspond to the robot inertia and friction. This avoids inertia shaping [Ott 08], enhances the robot behavior and makes the filling of the human collaborator more intuitive with respect to the robot reaction.

This control scheme was used in the stack of tasks as one of the basic tasks for building various behavior (walking while carrying a table, opening a door), see Fig. 7.1.

7.2 Vision

Reference paper [20] (see also [46, 13, 23, 38, 53])

The humanoid robot is definitely not the easiest platform to develop vision-based schemes. Actually, it is a very nice experimental platform to challenge a working scheme. The robot is at the same time a eye-in-hand (for localization) and eye-to-hand (for manipulation) system. Like other mobile robots, all the DOF are not measured by an encoder and the odometry is poor since the feet are slipping [Moulard 12d]. The camera is shaking due to the impacts during the walk. Eye-in-hand movements are not directly controllable since the head (following the COM) is oscillating around the main commanded direction [Davison 07, Dune 10]. At the same time, the vision greatly improves the behavior of the system, for obstacle detection



(a) (b) (c) (d)

Figure 7.2 *– Three example of the use of the camera. (a) Vision-guided grasping while walking. (b) Bi-manual vision-guided manipulation using the upper body. (c) Tracking of SLAM landmarks. (d) Motion generation on a dense reconstruction of an uneven floor [46].*

[Michel 05], floor reconstruction [46] (or similarly with proxi sensor [Chestnutt 09]) or to observe missing informations of the other sensors [Martinelli 12].

Visual servoing: During my stay at Tsukuba at the end of my PhD, we have worked with O. Stasse to demonstrate the capabilities of the humanoid robot to grasp an object while walking. This demonstration was realized like the force-based demonstrations presented in the previous section: the visual servoing is one of the basic tasks available in the action library, and we have combined it with a walking pattern generator and some posture tasks [38]. The complete scheme is finally obtained with a 2D eye-in-hand visual servoing to control the gaze, a 3D eye-to-hand visual servoing using the stereo camera pair to reconstruct the 3D information to grasp the ball, a Riccati-based pattern generator [Kajita 03] to walk and a posture task to prevent any rotation of the chest. Similarly, the same bricks where used (Master thesis of J-C. Renaud, LASMEA [70]) to perform bimanual manipulation using the upper part of HRP-2 (the feet were fixed on the ground) [53]. Fig. 7.2-(a) and -(b) give some typical executions with HRP-2.

Vision-based fine localization: One important issue for controlling the robot dynamics is to have an accurate estimation of the robot body orientation with respect to the gravity (vertical), the positions of the contact body (feet) and the model of the contacting surface. When the robot is walking on a perfect non-slippery horizontal rigid floor and if the flexibilities of the robot legs are neglected, all these information can be estimated from the joint encoders. However, for going on non-flat ground with elastic or slippery contacts, it is interesting to have proper measurement of all these effects.

The camera in the head can be used to accurately measure the position of the robot with respect to its initial position. Since the robot starts from a known vertical, this can be used to estimate its orientation with respect to the gravity. However, we need the measurement to be fast. For that reason, we have developed a real-time simultaneous-localization-and-mapping (SLAM) algorithm able to track the robot position at 60kHz [20]. See the project homepage [60] for details.

In a second time, the localization can be used to reconstruct a dense map of the environment. This map is then used to plan a first trajectory of the robot. It is used in [46] to validate the robustness of dynamics-based walking controller on rough terrain. The same

approach will be extended to plan the next footstep while avoiding the major holes of the ground.

7.3 Balance

The camera is able to reconstruct the robot position with respect to previous positions. The map helps to consolidate the accuracy. However, the orientation is subject to drifts and the camera is not able to accurately track alone the robot vertical. Moreover, it is not the proper sensor to estimate the contact model (surface orientation and stiffness).

Two other sensors can be used to bring the missing data. The inertial measurement unit (IMU) measures the angular velocity on one hand and the linear accelerations and gravitational forces on the other hand of the body it is attached to. When the linear acceleration are perfectly known (for example, when the robot does not move), the weight force gives the gravity orientation. When the acceleration is not known, the six measurements are not sufficient to make the gravity orientation observable.

A classical association is to use the camera to observe the missing information. The coupling IMU/Camera is very interesting [Martinelli 12]: the camera is slow ($<$100Hz) whereas the IMU is fast ($>$1kHz); the IMU quickly drifts whereas the camera does not drift while it stays in already-visited places. Moreover, the system camera+IMU is observable. A first fusion was performed in RT-SLAM.

We now tries to fuse the data coming from the force sensors in the ankles to estimate the elasticity in the robot feet and the ground characteristic. In parallel, F. Lamiraux is working on the stabilization of the feet elasticity and O. Stasse is working on the rough estimation of the floor slope using vision. Coupled with the developments of fast dynamics-based control law, all these works are going toward safe outdoor walking on unknown grounds.

Chapter 8

Partial conclusion

The objective of this first part was to construct a control structure to compose several simultaneous tasks, fast enough to be used as a robot controller, while ensuring a good behavior of the system in case of conflicts and taking into account all the robot modalities, in particular the dynamic effects and inequality-written objectives. As explained in Chapter 3, we have chosen to rely on a hierarchy to compose several tasks simultaneously. Among the objectives listed above, most of them are reached. In particular:

- For combining several tasks **simultaneously**, a very efficient hierarchical solver taking into account both equalities and inequalities have been proposed in Chapter 4.
- For combining several tasks **sequentially**, a first solution was proposed in Chapter 5, with some defects but that gives the directions of a complete solution.
- In Chapter 6, the **dynamics** of the robot was added in the problem.
- Combining the fast solver and the inverse dynamics, nearly real-time computations (**4ms per control cycle**) were achieved, that now enables us to study the application onto the real robot.

However, if we now have a working solution, two points remain to explored: the damping and the application on the real robot. Some perspectives are recorded below.

Damping: the goal is to ensure that, whatever the active tasks, the behavior of the system remains safe. Damping might seem a relatively minor aspects. Actually, it is very difficult explain its practical interest to theoreticians of numerical mathematics: we are deliberately searching to diminish the accuracy of our solver when the problem is too hard to solve for the robot capabilities. However, damping is very important if we want our solutions to be really used as the basic controller of the robot: if the activation of the tasks are decided by a non-expert human operator or by an automatic decision process, a good behavior has to be ensured.

The organization into a hierarchy is only one of the possible way to achieve this goal. When all the constraints are written as equalities, a solution that the whole robotics community is using is the so-called damping of the matrix inversion. The solution has poor theoretical properties but is very efficient in practice. We are not able yet to provide an equivalent solution when inequalities are considered. It might be possible to formulate the safe property of the robot behavior in the problem formulation (like bounding the norm of the solution or the variation of the control). Or it has to go with a rewriting of the solver operating model, to *disable* a given task when it is too difficult to be achieved. An alternative might be to rely on an external decision process, which would look at the problem conditioning number and remove the misbehaving tasks when the conditioning is too low.

Application to the real robot: In a sense or another, the application of inverse-dynamics control loops on the physical robot is one of the big challenges of the community today. At the mechatronics level, it is the purpose of the joint torque sensors of the DLR/Kuka LWR robot and the springs of the series-elastic or variable-stiffness actuators. We are trying to address this problem from the control point of view. This is interesting because the main efforts have yet been set on the hardware design rather than on the software control. Hardware solutions have now been proposed and software is the missing key. Our approach tries to speed up the computation in order to take into account the whole robot model and is therefore an interesting candidate for any hardware solution. In that direction, our experimental platform HRP-2 is equipped with both force sensors and springs. Both are located only in the end effector, which is limiting for many setups (*e.g.* hammering or throwing) but also by far sufficient in many other cases. Typically, the stabilization of the balance and the walk on a non-flat surface are challenging objectives on which we have started to work.

The execution on the real robot requires three aspects: we need to acquire the proper data, to extract the proper information from these data and to answer with a control modification as fast as possible. We have targeted most of our efforts on the last point, with some work on the second one. The first point requires to work more closely with the hardware design, to select the proper sensors and their connection with the actuators. In that direction, the recent work on the artificial skin (at Tsukuba [Mittendorfer 13] or in the Factory-in-a-Day FP7 project) and on variable-impedance actuators (with B. Tondu and O. Stasse) is very promising.

In addition to this medium-term perspectives, two other directions deserve to be mentioned.

Conic problems: We have only worked with quadratic cost functions, *i.e.* linear constraints. For example, it is not possible to impose a bound on the norm of the parameter, which would be written as a conic constraints. For the humanoid robot, it is not possible to take into account the exact friction cones, that are, namely, conic constraints. The extension to the presented work for other types of constraints will be necessary at some point. The same questions of hierarchy, continuity or damping are still relevant for more-generic classes of numerical problems. Conic problem is the first to consider, because of its similarity to quadratic problem. However, conic solvers do not rely on active-search algorithms, which is the solution that we have used as the basis of all our work in Chapter 4. The HQP solver

might still be exploited, as most of the other resolution schemes rely on an internal linear solver.

Preview capabilities: By construction, the task-function approach realizes an instantaneous linearization of the task function and therefore brings back the problem to a linear formulation. However, this kills the view on the future. Typically, on the humanoid robot, a view on the future is necessary to consider the robot balance. This cannot be simply done with the instantaneous linearization. We then rely on a trajectory over the next few instants, that is computed by another system called the walking pattern generator and that ensure that the behavior computed locally lead to a relevant behavior in the next future.

Walking is only the most exemplary case where a preview window is necessary. We have yet no good solution to systematically integrate the preview in the considered framework. A solution would be to discard the instantaneous linearization and to consider the whole system over the preview window. However, this leads to some strong non-linearities, which are first computationally expensive but are also very sensitive from a numerical point of view. Further elements about this topic are given in Chapter 10. With today knowledge, the artificial decoupling between on the first hand a preview walking generator to handle the future and on the other hand an instantaneous linearization to handle the whole-body execution is much more efficient (even if less optimal in term of movement) that the resolution of both coupled. A systematically way to generalize this approach to other case study is still a perspective.

From Part I to Part II. The task-function approach defines basic bricks that characterize elementary motions of the system, robot or avatar. Part I gave some solutions to combine this bricks into a more complex motion. We did not discuss yet how to select the proper bricks and in which order to assemble them. The next part will focus on this topic. The first solution is to select them "*by hand*", using the knowledge of a robotics expert to design the motion. By providing the basic sensory-motor loops, the framework presented here is a very powerful tool for such a robotics engineer **to program the robot motion**. Another solution is to use **planning techniques**, combined with a model of the world, to decide which sequence of tasks to use. Finally, the tasks can be sequenced following a model **extracted by observation**. This last direction directly leads to learning. Each of these three approaches will be described in the next part.

Part II

Emergence of a motion semiotics

Chapter 9

A programming language

The task-function framework built in the previous part offers some basic bricks of motion. It can be seen as a motion vocabulary with a rough-and-ready composition syntax. This part will show how this vocabulary can help to create some complex behavior. Each of the three chapters of Part II focuses on a different approach: composition by explicit programming, by automatic planning or by imitation. The last chapter, number 12, will present two demonstrations using the proposed concepts.

First of all, the task function is a very expressive tool to help an engineer to design a specific motion on the robot. We have built a software called *StackOfTasks*. It implements the mathematical components introduced in the previous part (some classic task functions, a hierarchical solver, inverse dynamics, continuity, etc) along with a scripting encapsulation that eases the construction of complex motions. This chapter describes the originality in the software architecture and quickly lists the available features. Throughout the chapter, a basic example, described in Section 9.1, is used to illustrate the concepts.

9.1 Overview

Reference paper [30]

The objectives of the software are the following. First, in order to reach real-time control, we need to embed some efficient piece of software (typically, C++, Matlab would not be enough). On the other hand, we would like to be able to prototype robotics application quickly. Finally, we want to reduce the amount of know-how required to create a robotics application.

To reach these objectives, the software is architectured around a computation graph, which can be accessed by an versatile scripting language and whose nodes are efficiency implemented. Each node of the graph is called an *entity*. The links between entities are called *signals* and represent the data produced by an entity and consumed by one or several other entities. The signals define a dependency graph, which is a partial order of the nodes. One of the maximum

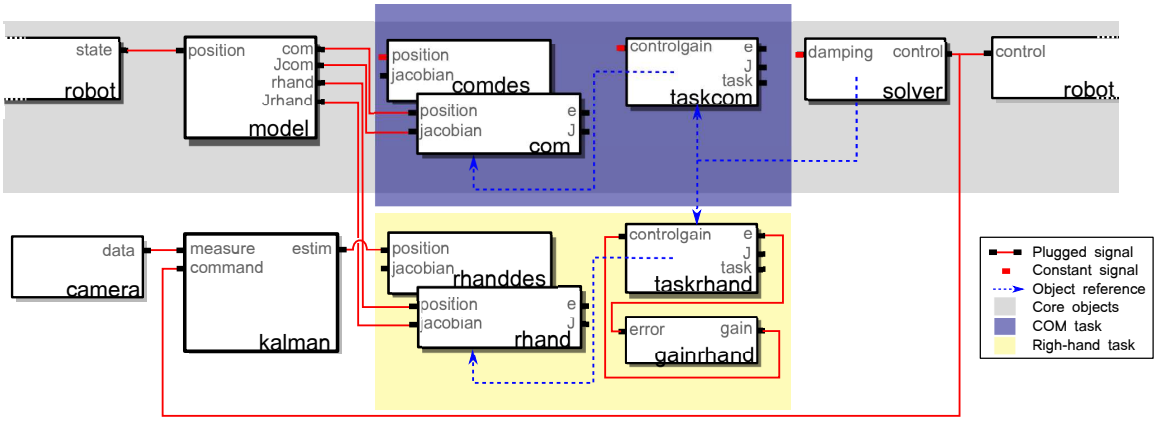


Figure 9.1 – *Typical example of computation graph. Here is implemented a simple behavior of visual-guided grasping with COM regulation to ensure the balance.*

of the graph is typically the control law, whose computation triggers the recomputation of all the lower computation nodes.

An example of graph of computation is given in Fig. 9.1. This graph codes a vision-guided grasping motion on a humanoid robot. Two tasks are defined to control the COM and the right hand. The COM is simply regulated to a fixed position with a constant gain. The right-hand is servoed to a 3D position measured by a camera (*e.g.* the position of a ball). The gain is varying depending on the norm of the error. To improve the tracking accuracy, a Kalman filter is used to correct the data coming from the camera using the known robot control. The direct model functions (COM and hand positions, Jacobians) are computed by a dedicated entity called *model*. The solver finally uses the references defined by the two tasks to compute the control law, which is fed to the robot.

In the following sections, we give the keys to understand this graph and extend it. The computation graph is first explained in Section 9.2. The basic task-function features are then shortly described in Section 9.3. Finally, the available packages of the software suite are listed in Section 9.4 and some examples of use are given in Section 9.5.

9.2 Dynamic computation graph

The boxes of the diagram presented in Fig. 9.1 are called the entities. Each entity has two kinds of interfaces: the signals and the commands.

The signals are synchronous interfaces that define the computation graph. Producer slots are called *output* signals. Consumer slots are called *input* signals. An output signal can be plugged to one or several input signals. Input signals can also be set to a constant value (*e.g.* the `controlgain` of the COM task is constant).

Output signals can be recomputed by a function callback or triggered by an external event. Recomputation by a callback is the most classical case. In the given example, all but the robot state and the camera data would be of that kind. Callback signals are associated to a function and are given a list of input signals of the same entity they are depending on. These dependencies are not made explicit on the figure to keep it readable. For example,

the `estim` output signal of the `kalman` entity depends on both the `measure` and the `command` input signals. On the other hand, event-triggered signals are changed by an external system (for example, data coming from the sensor or user-driven modification).

Any signal is tagged with the control-cycle number of the last recomputation and keeps a copy of the recomputation result. When an output signal is accessed, it is recomputed under the following inductive conditions:

- If the current control cycle number is higher than the recorded one.
- And if at least one of the signal it is depending on is ready to be recomputed (following the same recursive rule) or is tagged with an earlier control-cycle reference.

The control graph is a loop by definition of the closed-loop control. It should be displayed on a cylinder rather than on a plane. However, it is not recomputed continuously but discreetly. To represent it and to discretize the computation, the loop is broken at an arbitrary point. We have chosen here to break it at the robot level. At each control cycle, the robot operating system modifies the state from the encoders measurements and access to the `control` signal. The dependency evaluation flows down to the `state` signal, which is ready for an update. This triggers an upward cascade of recomputation until the control if finally recomputed. Interestingly, the part of the graph corresponding to the right-hand task is not recomputed if no new data has arrived from the camera. This spares some useless recomputations.

Desirably, all the computation dependencies should be expressed through signals. However, for historical or simplification reasons, some dependencies are implicit. Typically, an entity can have a pointer on another entity and directly use its signals. For example, the `solver` entity maintains a list of the active tasks and uses the `task` and `jacobian` output signals of these tasks to recompute the control law. These hidden dependencies are denoted by dotted lines on the graph.

The commands are asynchronous interfaces that are typically used to build the graph structure and to parameterize the entities. For example, a task is added to or removed from the solver by a command. The parameters of the adaptive gain of the right-hand task are defined by a command. The files defining the robot model are set by a command.

Scripting and abstraction: The architecture is build around two conflicting goals: being real-time and being able to prototype quickly. To achieve both, a classical software engineering pattern is used, by splitting the design into two levels: the computation part in C++, which needs to be efficient; and the graph in python because this is what is "application dependent" (it involves a lot of disposable code we do not want to spend time writing). This is a very common approach used by other software (like the 3D modeling software Blender with Python) and some languages are even designed explicitly for this (Lua [Ierusalimschy 96], which encountered a large success for game scripting). Other robotics middleware even incorporates this at a very low-level (like the robotics language Urbi [Baillie 04])

All the graph (entities, signals and bound functions) are defined in C++, which is compiled but computationally efficient. The objects are available through the entity, command

and signal abstraction. They can then be manipulated by the mean of an external scripting language. A python interface has been developed that gives access to the creation and destruction of entities, signal plug, recomputation and command calls.

To ease the extension of the language, any new entities can be developed in an external library and introduced during the execution by mean of a dynamic plug-in. A typical execution is then set in three phases:

- The needed functionalities, embedded in the corresponding plug-ins, are loaded.
- The entities are created, parameterized by mean of commands and their signals are plugged.
- The execution is triggered by externally calling one of the entities (typically, the `robot` entity).

The execution can be modified by changing the signal graph (unplug or replug some signals) or the entities configuration (call of a command). Typically, task sequencing is achieved by calling the `push`, `remove` and `swap` functions of the hierarchical solver.

The graph functionalities are implemented in the package dynamic-graph (see Table 9.1). The links with the python language are implemented in dynamic-graph-python.

9.3 Motion implementation

The computation graph described above is generic and, even if originally designed for control, could be used for various frameworks. The task function is specifically implemented in the `sot-core` package. It is based on the implementation proposed in the ViSP library [Marchand 05]. The task implementation is decomposed in two parts: the function itself, called *feature* on the one hand; and the computation of the reference vector $\dot{e}^*$, called task, on the other hand.

The feature is an abstraction that provides two output signals: `error` (which is the function of the robot current configuration and other environment parameters) and `jacobian` (which is the derivative of this function with respect to the robot configuration). The `error` is most of the time computed as a difference between a measurement and a reference value. Most of the time, the measurement is an input signal of the feature entity itself, while the reference is an input signal of a joint feature entity called the desired feature. For example, in the illustrative graph of Fig. 9.1, the `com` feature comes with a `comdes` feature that specifies the reference COM position. Similarly the right-hand feature `rhand` comes with a `rhanddes`, whose input is coming from the camera.

The basic feature proposed in the `sot-core` package is called `FeaturePoint6D` and characterizes the distance between two body placements. It is typically used to bring the robot end effector to a desired placement (*i.e.* position and orientation). It is typically used to implement the right-hand task in the example. Variations are also proposed (distance between two robot bodies, distance of a point to a line, distance between two vectors of the space). Finally, a so-called generic task function is implemented, typically used by the COM task (whose Jacobian is directly computed from the robot model).

Package	Git repository	Description
dynamic-graph	jrl-umi3218/ dynamic-graph	Define the entity and signal abstraction, implement the dependency graph and the dynamic loading functionalities
dynamic-graph-python	jrl-umi3218/ dynamic-graph-python	Bind the dynamic-graph abstractions with the python scripting language.
sot-core	jrl-umi3218/ sot-core	Implement the basic task-function functionalities: control feature, tasks, inverse-kinematics solver, simple-integrator robot, etc.
sot-dynamics	jrl-umi3218/ sot-dynamics	Implements the *StackOfTasks* bindings with the model computation library `jrl-dynamics`.
sot-pattern-generator	jrl-umi3218/ sot-pattern-generator	Implement the *StackOfTasks* bindings with the walking pattern generator `jrl-walkgen`.
soth	laas/ soth	Implement the HQP presented in Chapter 4 and [4].
sot-dyninv	laas/ sot-dyninv	Implement the hierarchical operational-space inverse dynamics solver.

Table 9.1 *– List of the StackOfTasks packages. The Git Hub repositories are available on the cloud on the server github.com (access to the http interface of each repository by adding the prefix https://github.com/). The whole project, supported jointly by LAAS and JRL-Japan, will soon move to the dedicated GitHub repository https://github.com/stack-of-tasks .*

The task is an abstraction that provides three output signals `error`, `task` and `jacobian`. A list of features are internally maintained (managed by an *add* and a *remove* commands). The `error` and `jacobian` are simply computed by stacking the errors and the Jacobians of the features of the list. The `task` signal computes $\dot{e}^*$ from e.

The default implementation of $\dot{e}^*$ is the proportional $\dot{e}^* = -\lambda e$. The gain is an input signal and can then be synchronously recomputed, which gives a simple solution to obtain an adaptive gain. Other vector fields are also provided.

Basic mathematical toolkit: A simple hierarchical solver (3.26) is implemented (no inequalities). A simple integrator robot (the input $\dot{q}$ is integrated using an Euler integrator to obtain the configuration q) is provided. A logger is implemented to record the signal data flow. Many other minor tools (such as simple binary operators, rotation algebra, a Kalman filter, etc) are also implemented to fasten the development of experimental graphs. The ultimate goal is to provide all the tools to build your control law by just reusing basic building blocks, without having to write entities anymore.

9.4 A whole humanoid-robot framework

The *StackOfTasks* is part of the humanoid path planner (HPP) software suite. Several extensions are proposed and are quickly listed below.

Model direct functions (sot-dynamics): This package provides the direct functions of the robot model: direct geometry (computation of the robot-effector positions and orientations, position of the COM), direct kinematics (Jacobian of the robot-effector positions and orientations, velocity of the effectors), and dynamics (recursive Newton-Euler Algorithm (RNEA) to compute the joint torques, composite-rigid-body algorithm (CRBA) to compute the generalized inertia matrix, ZMP).

The input signals of these entities are the robot joint positions, velocities and accelerations. The position and orientation of the robot in the space (free-floating placement) and corresponding velocity and acceleration can be independently provided. The position information alone is used by the geometric computations. Velocity is additionally needed by the kinematic computation. Acceleration is only needed by the dynamics computations.

The output signals are the COM, its Jacobian, the ZMP, joint torques, dynamic drift (sum of gravity and Coriolis forces) and generalized inertia matrix. A command triggers the creation of a pair of signals outputting the position and Jacobian corresponding to any robot-body central point. We cannot yet compute the Hessian of the geometric quantities nor the square root (Cholesky decomposition) of the inertia matrix (both quantities might be computed from the model and are needed for inverse dynamics).

All the algorithms are implemented in the package `jrl-dynamics`. They mainly correspond to the formulation proposed by Featherstone [Featherstone 08]. Alternatively, automatically-generated optimized code can be used. The algorithms are executed in dedicated symbolic computer software (such as Mapple), which unroll the algorithm computation trees and produce an automatic code dedicated to one specific robot model. This approach is satisfying in terms of performances but is more difficult to maintain and distribute on a software point of view. In [67], a very important effort of advanced programming techniques, led by O. Stasse, was done to perform quasi-symbolic code optimization using meta-template programming (in C++). The performances are close to the execution times obtained with dedicated automatically-generated code but all the optimization are done only at the compilation of the object code, while no dedicated source code is generated.

Generation of walking patterns (sot-pattern-generator) Several pattern-generation methods are proposed in the package `jrl-walkgen` by O. Stasse. It is wrapped in the *StackOfTasks* framework. The main entity produces COM trajectory giving footprint positions [Kajita 03], enables modification of the footprint stack [Morisawa 07], or automatically computes the next footprint positions during the COM-trajectory optimization [Herdt 10]. Other entities compute the flying-foot trajectory and synchronize the task sequence with the footsteps.

Hierarchical solver (soth) A generic implementation of the hierarchical solver [4] is proposed in this package. The implementation is done outside of the *StackOfTasks* framework, based on the Eigen numerical software [Guennebaud 10].

Efficient inverse kinematics and dynamics (sot-dyninv) An inverse-kinematics solver and an efficient (condensed) inverse dynamics solver are built based on the `soth` solver and the *StackOfTasks* framework. A inequality-based kinematic task is also proposed, along with a proportional-derivative dynamic task and an inequality-based dynamic task.

Many python scripts are also given, as examples of robot behaviors that can be implemented with the *StackOfTasks*.

Simple 3D rendering interface (robot-viewer) Robot-viewer is a simple visualization interface implemented by D. Dang. Robot models and 3D objects can be loaded from VRML model files and moved from another application or computer by a simple XML-RPC or CORBA interface. Robot-viewer is not a simulator. It can be used to picture the state of a simulator and the state of the physical robot during a real execution. A simple XML-RPC and CORBA client is also proposed in python.

Inclusion in the ROS framework ROS (Robot Operating System) is a development and distribution framework dedicated to robotics led by the USA company Willow Garage [Quigley 09]. It provides the basic facilities for inclusion of algorithms on real robot platforms (communication middleware, data log plots, camera drivers, simulator, etc.) along with many various algorithms (SLAM, object recognition, planning, etc.).

The *StackOfTasks* software was added as a package of ROS by T. Moulard, who also wrote the ROS drivers for the HRP-2 robot. In particular, ROS is used to provide the communication interface with other computers (for example on the HRP-2 to communicate with the computer that acquires the camera images). In cooperation with G. Manfredi (RIS/LAAS-CNRS), a parser was integrated to read robot models using the URDF format and the PR-2 [Wyrobek 08] robot model was included in the *StackOfTasks*.

9.5 Examples of use

The *StackOfTasks* software has been used in the last four years in four major robotics French laboratories (LAAS, LIRMM, JRL-Japan and INRIA Grenoble). It was mainly used with the HRP-2 robot. Some of the demonstrations built upon this software suite are quickly presented below. A visual summary is given in Fig. 9.2.

- **Robot@CWE [3] (JRL-Japan):** Complete integration setup, detailed in Chapter 12.1. This is the final demonstrator of the FP7 IP Robot@CWE project.
- **Grasping while walking [38] (JRL-Japan):** Grasping a ball with visual guidance while walking. First demonstration with the presented framework during the end of my PhD.
- **Open the fridge [34] (JRL-Japan):** The robot open a fridge to grasp a can. The motion results from an optimized task sequence, as detailed in Chapter 10.2. The motion was implemented by F. Keith during his PhD thesis.
- **Haptic communication [Bussy 12] (JRL-Japan):** Study about the haptic (non verbal) communication between two human partners during a task implying physical

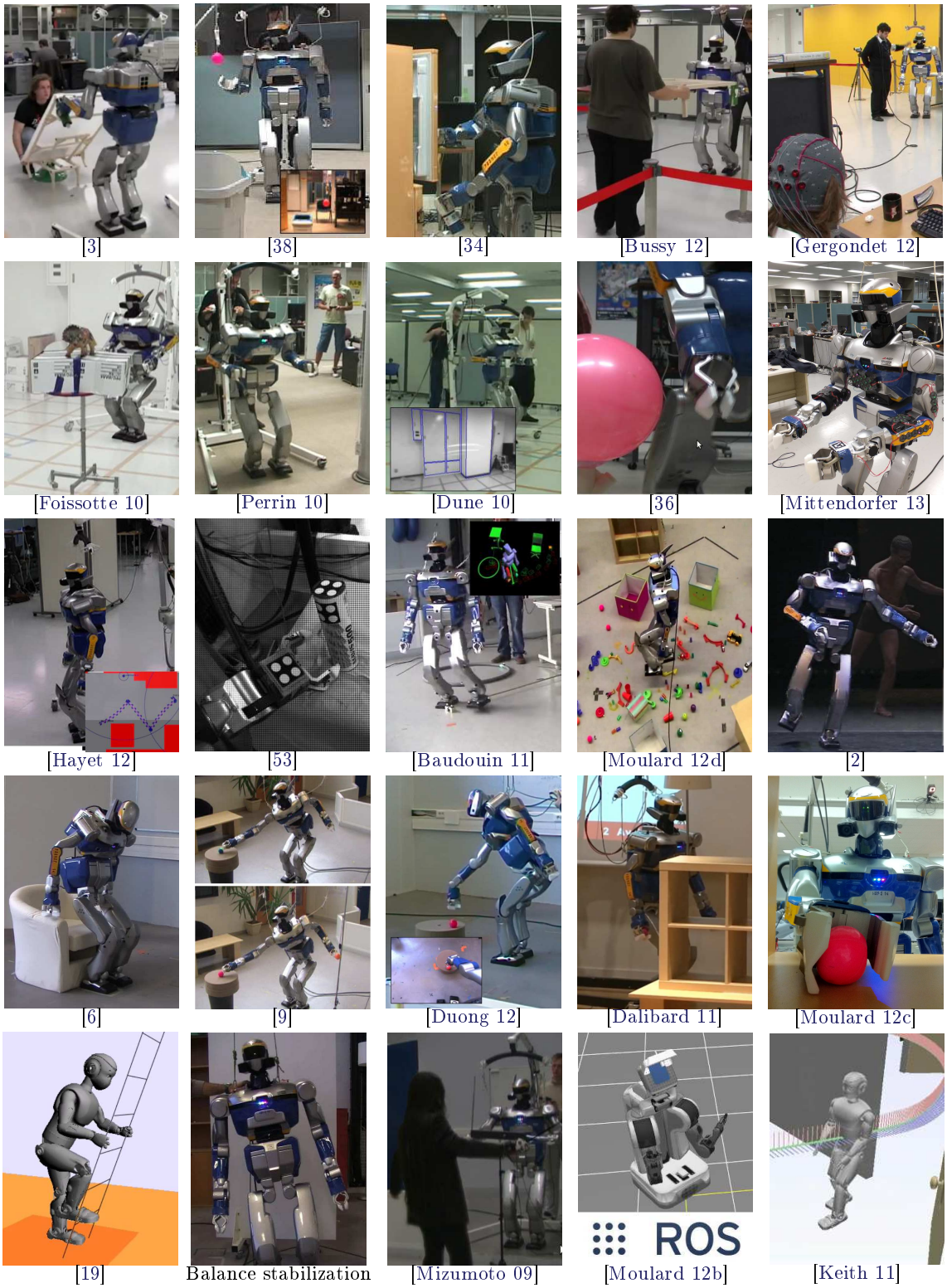


Figure 9.2 – *Some of the demonstrations performed over the last four years with the StackOf Tasks software, mainly on HRP-2, by four major robotics French laboratories (LAAS, LIRMM, JRL-Japan and INRIA Grenoble).*

collaboration. The resulting strategies are applied to drive the humanoid robot decisions. The motion was programmed by A. Bussy during his PhD thesis.

- **Brain computer interface [Gergondet 12] (JRL-Japan):** The robot is driven by symbolic orders that are selected by a brain-computer interface. The orders are converted into tasks that are applied by the robot. The demonstration was developped by P. Gergondet during his PhD thesis. It is one of the main demonstrators of the FP7 IP VERE project.

- **Active vision [Foissotte 10] (JRL-Japan):** The robot autonomously selects the most relevant poses to improve the 3D reconstruction of an object. The built model is then used for the "Treasure hunting" project, led by O. Stasse [Saidi 07]. The active-search movements have been developed by T. Foissote during his PhD thesis.

- **Joystick-controlled walk [Perrin 10] (JRL-Japan and INRIA):** First application of the "joystick-drive" walking pattern generator: the COM trajectory is the closest to the velocity input given by the joystick [Herdt 10]. The footprints are computed during the COM trajectory optimization and bounded to stay inside a security zone computed during a learning process [Stasse 09].

- **Visual servoing of the walk [Dune 10] (JRL-Japan):** Similarly to the previous item, the robot COM and footprints are computed by model-predictive control. This time, the robot velocity is input by the camera feedback, following a visual-servoing scheme.

- **Obstacle avoidance [36] (JRL-Japan):** The obstacles are locally taken into account in the control scheme. The task used to prevent the collision is based on a smooth body envelop [Escande 07].

- **Skin for humanoid robots [Mittendorfer 13] (JRL-Japan with TUM):** Several tactile and proximetric sensor cells are attached to the HRP-2 robot cover. They are used to guide the robot and teach by showing how to grasp various classes of object using whole-body grasps.

- **Pursuit-evasion planning [Hayet 12] (JRL-Japan with CIMAT):** A navigation trajectory is computed while taking into account the robot visibility constraints. The plan is then executed using the walking pattern generator and the proposed whole-body resolution scheme.

- **Bi-manual visual servoing [53] (LAAS with LASMEA):** The head and both arms are controlled following a 2D visual feedback.

- **Fast footstep replanning [Baudouin 11] (LAAS and JRL-Japan):** Using fast feasibility tests, the footsteps leading to a goal position are recomputed on the flight to track modifications of the environment. The footstep plan is then executed by inverse kinematics.

- **Careful steps through a kidroom [Moulard 12d] (LAAS):** A footstep plan is computed into a very constrained environment. The flying-foot position is servoed

during the execution based on visual feedback. The demonstration was realized by T. Moulard during his PhD thesis.

- **Yoga dance [2] (LAAS):** The robot motion is computed from motion-capture dynamic motion of a human dancer. Most of the work has been done by O. Ramos during his Master thesis. The demonstration is detailed in Chapter 12.2.
- **Sitting into an armchair [6] (LAAS):** The motion illustrates multi-contact capabilities of the generation algorithm. It is executed in open loop by the robot. This motion was developed by L. Saab during her PhD thesis.
- **Motion recognition [9] (LAAS and JRL-Japan):** The task-function approach is used to describe an observed motion. The approach is detailed in Chapter 11. It was developed by S. Hak during his PhD thesis.
- **Visual footstep planning and control [Duong 12] (LAAS):** The framework is used to execute a footstep plan computed by homotopy [Kanoun 11b]. The footstep trajectory is computed at the medium rate of 3Hz. The footsteps and robot posture is then sent to the control and performed in real time. Both algorithm loops are closed on the camera feedback. This motion was developed by D. Dang during his PhD thesis.
- **Small-step controllability [Dalibard 11] (LAAS):** A trajectory of the humanoid robot sliding on the floor is computed by a sampling-based motion planner. It is then shown that this trajectory can be executed with no other condition by a walking robot. The complete trajectory is finally applied by controlling the walk using the *StackOf Tasks*. This motion has been developed by S. Dalibard during his PhD thesis.
- **SLAM-based robot control [Moulard 12c] (LAAS):** The robot localizes itself using a visual sparse map. The localization error is back fed to the control that corrects the position with respect to an input plan. The robot finally grasps a ball after several steps. The demonstration was developed by T. Moulard during his PhD thesis.
- **Climbing a ladder [19] (LAAS):** Application of the multi-contact inverse-dynamics motion generation to the humanoid robot Romeo model.
- **Balance control (LAAS):** Developments toward outdoor walking, by F. Lamiraux.
- **Audio-based control [Mizumoto 09] (LAAS with SIPG Okuno Lab)):** Developments for the standing HRP-2 of an audio-based control, originally developed for a sitting HRP-2. The hand position is servoed to adjust the sound of the Theremin music instrument. The tempo is then adjusted by a conductor using the motion capture to track the baton.
- **ROS bridge [Moulard 12b] (LAAS):** The framework is bridged to the middleware ROS [Quigley 09]. The middleware is used to import data coming from other processes or other computers (vision data, user commands, etc). The PR-2 model is included in the framework.
- **Navigation planning and execution [Keith 11] (INRIA):** The robot Romeo navigates into a virtual home environment and finally performs a manipulation task. Final demonstrator of WP7 of the FUI Romeo project.

A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: the Stack Of Tasks

N. Mansard, O. Stasse, P. Evrard, A. Kheddar

IEEE International Conference on Advanced Robotics (ICAR'09) [30]

Context:
This paper was written after my post-doc in JRL to present the open-source software *Stack Of Tasks* written in collaboration with O. Stasse, P. Evrard and A. Kheddar.

Motivations:
A inverse-kinematics based movement on the robot contains many basic routines that are reusable but have to be written on a very efficient manners, and topical implementations that does not require a high computational efficiency but is more difficult to reuse from an application to the other. The software is proposed to maximize the efficiency of the kernel routines and allow an easy integration of these routines to produce an actual robot movement.

Approach:
The software is built as a mixed between C++ efficient routines, and a scripting language to construct the robotics scenarios. The control scheme is assembled using a graph structure (using a design pattern similar to MATLAB/SIMULINK) that links together the C++ routines. The basic features are then proposed to realize an inverse kinematics scheme: robot model computation (direct geometry and kinematics, Jacobians, generalized inertia matrix, etc), classical task functions (end-effector positioning, COM, gaze, etc) and a hierarchical inverse kinematics solver.
The software is then distributed with a LGPL license.

Results and contributions:
The software was used to build several demonstrations with HRP-2. It is the basic software for most of the robot experiments at JRL-Japan and in the Gepetto group. It was also used at INRIA, LASMEA and LIRMM.

Limitations and perspectives:
A second version of the software has now been proposed. Python is used as the scripting interface and CORBA was replaced by ROS for computer networking.
The control graph is supposed to be run on a real-time operating systems and does not provide synchronization mechanisms in itself. The SIMULINK-like design pattern might be limiting for reuse of code pieces to other contexts. Even if generic, the software was only used for humanoid robots and was not validated e.g. for fixed or mobile manipulators.

Chapter 10

Task sequencing

The previous chapter gave a solution to program a robot motion by a sequence of tasks. We now look at a solution to automatically decide the data that are needed to specify the motion. When reasoning about a motion (for planning for example), the whole spatio-temporal trajectory should be considered[1]. The motion planning (both for searching for an initial solution [Barraquand 92] or to optimize a given trajectory [Diehl 09]) is a very complex problem, subject to the question of the existence of a solution and of many local minima. This is the "good" mathematical way to build a motion. However, the computational cost is yet much too expensive to hope to use it inside the robot control loop. Moreover, if it is used as a planner (*i.e.* to guide the controller by computing the desired motion as a much slower rate), it provides a trajectory in the configuration space, that is straight-forward to track with a fixed manipulator robot but can be more complex to follow with a mobile robot. The question behind this chapter is to know how to execute with a closed-loop controller a trajectory computed by a motion planner, or more generally how to bridge the gap between planning and control. In this objective, the task sequence can be used as a trajectory representation that makes the link between these two components: we show in this chapter that the task sequence can be used for planning, while the first part of the document has shown that it is a very efficient way to control the execution.

In order to introduce this chapter, a quick summary of the work on optimal control is first presented in Section 10.1. Most of the chapter is then condensed in Section 10.2, where we propose a solution to optimize a trajectory represented by a sequence of task. Contrary to the previous chapter, this chapter does not propose a complete method answering to a given problem. It is rather a first step toward the prospective, as discussed in Section 10.3.

[1] In humanoid robotics, the complexity is very often broken by decoupling the spatial dimension (dealt with the instantaneous linearization of the stack of tasks) from the temporal dimension (dealt with the inverse-pendulum based pattern generator).

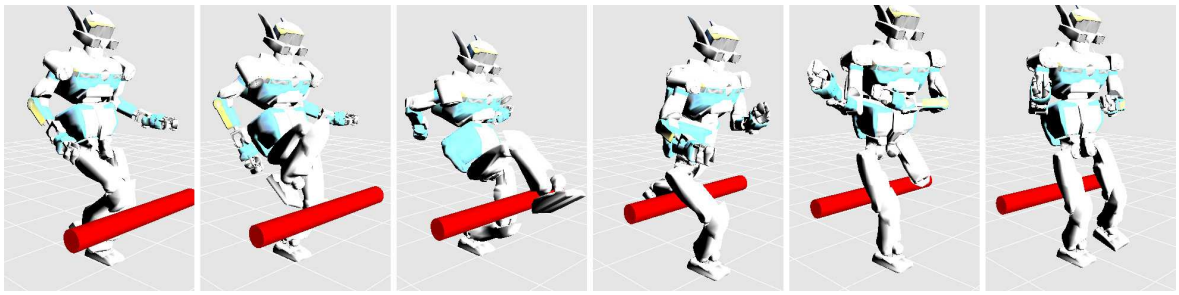


Figure 10.1 *– Stepping over a large obstacle using optimal control. Results from K. Koch [15].*

10.1 Optimal trajectories

Reference paper [15]

This section quickly reports an on-going collaboration with K. Mombaur and K-H. Koch from IWR Heidelberg to implement whole-body optimal trajectory on the HRP-2 for walking or other movements. The problem is modeled by an infinite non-linear problem (infinite number of variables, infinite number of constraints), where the variables are the robot configurations and derivatives during a time interval and the input control function on that interval. The control is typically reduced to a finite-dimension generating family defined by a given basis of functions (*e.g.* splines or step function on one element of a sampling grid), while the configuration is obtained after integration of the robot dynamics. The integration is subject to contacts with the environment. For walking, multiple contact phases with impact and/or continuity constraints are set up. Here, the contact phases are given at the problem definition. The typical robotics constraints are then added (obstacles, joint limits). Finally, the cost to be optimized is selected among a set of possible function (minimum torque, maximum forward velocity, minimum impact, etc.) or composed as a weighted sum of them.

Since the integration of a given control on a humanoid model is unstable (the robot is very likely to fall), a multiple-shooting numerical solver is used [Bock 83, Stoer 02]. Such solvers are typically designed to work with unstable systems such as rockets. Basically, the problem considering the whole trajectory is transformed into several independent problems on pieces of the whole time interval. These pieces are linked one to each other by continuity constraints. The gradient matrices corresponding to the multiple-shooting formulation are much larger than with a single-shooting formulation but are sparse, typically with a strong diagonal. A condensation phase can bring the large sparse problem to a dense problem whose size is equivalent to the single-shoot formulation but has a much better stability. An implementation (not free) of such a solver is described in [Leineweber 03]. It was used for manikin animation, for example for running in [Schultz 10]. A very complete description of the method summarized in this paragraph is available in this last paper. An example of use to step over the largest possible obstacle is given in Fig. 10.1: the height of the obstacle is the optimized objective assigned to the solver.

There are two directions that can come from this approach. A first one is to see the non-linear trajectory optimization as the future of the work described in the first part, in particular in Chapter 6: up to now, we have instantaneously linearized the system, approximating the

robot future by a constant. Of course, from the studies *e.g.* about walking, we know that this simplification cannot handle all the modalities of the robot. Optimal control with the whole robot dynamics is a Graal that is not close to be reached. Three difficulties have to be solved before applying it to actually control the movement of humanoid robots:

- If considering the trajectory, the linearity is lost even if approximations can be found to keep it [Pham 12].
- The number of variables drastically increases: it is multiplied by the number of timesteps to be considered, typically more than 100. Some works show that this complexity could be handle by a meticulous and dedicated implementation [Tassa 12].
- A last stumbling block is the relevance of the output trajectories: we have experimentally observed with the work on HRP-2 that this additional redundancy offered to the solver (temporal redundancy, in addition to the already understood motion and actuation redundancy) might be difficult to handle.

A second direction is to see the optimization phase as a planner that selects one motion, during an off-line phase (or on a lower update rate). This motion has to be followed by the robot. There is then some reflection to do to obtain a trajectory that is described in a way that eases the work of the controller. The next section can be seen as a trajectory optimization, where the task control approach is used instead of dummy piecewise constant controller. From such a trajectory representation, we can hope to obtain a trajectory with a rich semantic to be more easily applied by the control.

10.2 Optimization of sequences

Reference paper [34] (see also [62])

The previous section gives a very nice example of the typical result of a robot planner: the output is a joint trajectory which can be directly replayed by the robot using only the joint encoder feedback. If the motion replay fails for any reason (change in the environment, imperfect robot motion, imperfect map, etc.), the plan generally has to be recomputed again. It is indeed a difficult problem to find how to generically follow in closed loop a joint trajectory computed by a planner [Moulard 12a].

For example, consider grasping a ball while maintaining the robot balance in an obstacle field. If the ball moves, what compensation should be applied on the COM? What is the *cause* of the knee joint motion, and consequently what is the sensor information that should be used to correct it if a deviation appears in the plan: the proximeters (obstacle), the camera (ball) or the gyro (balance)? Or is this perturbation meaningless and caused by a random numerical effect inside the planner?

The trajectory computed by the motion planner is often given in the configuration space and has a very poor semantics. In particular, the planner gives no *explanation* about the trajectory it has returned while such additional descriptions would be needed to select the proper way to regulate the motion execution. For fixed manipulator robots, the configuration space is the logical way to represent the trajectory [LozanoPerez 83]. When the action is explicitly referred to a given sensor (for example "*place the embedded camera in front of a target*"), a sensor-based task space can be chosen to more appropriately represent the trajectory [Mezouar 02].

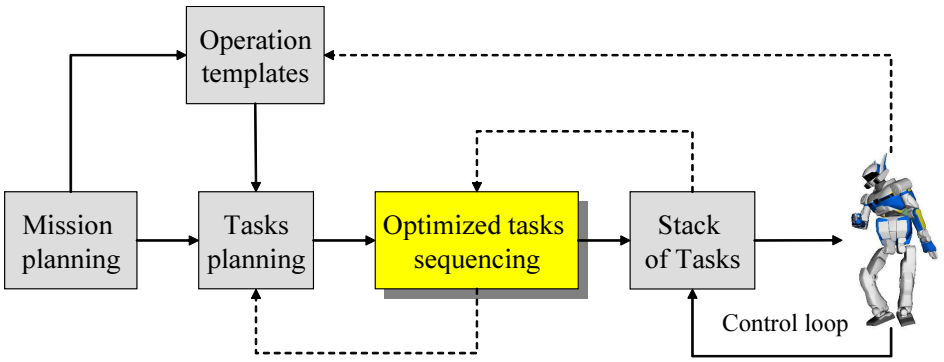


Figure 10.2 *– From mission planning to execution with the task-sequence optimization module*

The proper landmarks for one sensor can be also automatically selected for longer range of motion [Malti 11]. However, the trajectory, expressed in the configuration space or into a isomorphic space, may not be the easiest way to specify the movement: for example, tracking a configuration trajectory with a mobile robot is not straightforward [DeLuca 97] while a rougher motion description would ease the control [Segvic 09]. For humanoid robots, the motion may be even more complex to specify as it requires several modalities (several sensors, several feedback controls) with various levels of action models and environment models.

The idea here is to use the tasks (in the definition used in the previous part and last chapter) as basic bricks upon which both the plan and the execution are built. The task, which has been seen **as a numerical controller of the execution** up to now, is seen **as a symbol in the reasoning** part. The trajectory is then expressed by a sequence of tasks that the reasoning modules manipulate while computing the robot motion [Dantam 13].

However, the task by itself is not sufficient to define the robot behavior and thus to set up the control. It needs to be configured with some numerical specifications, typically the value of the gain, the time of start, etc. The task is therefore a proto-symbol, close to the sense defined in [Inamura 04]: we can reason on it like if it was a symbol but without forgetting that numerical details are embedded into it.

10.2.1 Planning and replanning

From the planner, we suppose that a sequence of symbols is given. The symbols are then refined by an optimizer (that acts as a scheduler), enriched and upgraded to proto-symbols. The resulting plan can be directly executed in closed loop by the robot. An overview of the treatment flow from planning to execution is given in Fig. 10.2. This flow is detailed below. During the description, the example given in Section 10.2.3 is used to materialize the concepts: the robot has to open the fridge with the right hand and to grasp an object inside the fridge with the left hand.

Planning: In its general meaning, task planning [Ghallab 04] consists in building the appropriate sequence of tasks to realize a given mission. In other words, it consists in choosing in a given pool of doable tasks (called know-how) the adequate set of tasks and ordering them to achieve the given mission. In such planning, the task is defined as a symbol that can be translated into an action or a set of actions to be concretely realized and usually comes with

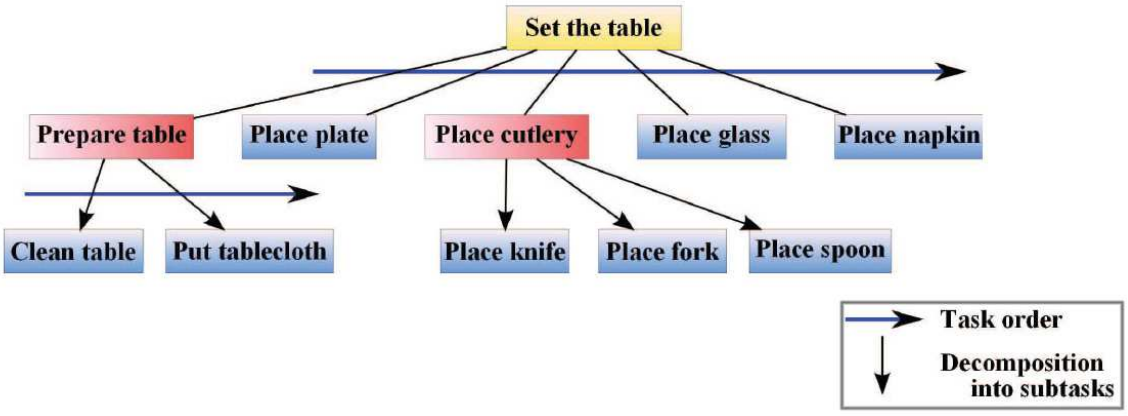


Figure 10.3 *– Example of mission mission decomposition (set the table) into a task plan. Each leaves of the mission tree may be interpreted as a task as defined in Chapter 3. Figure from F. Keith thesis [62].*

a set of preconditions. A large majority of planners directly work with tasks from the pool. More powerful hierarchical planners (Hierarchical Task Networks) use task complexes with a descending approach from high level tasks down to low level ones. A survey can be found in [Smith 00].

At the end of the planning phase, we have a set of sequence of symbols to be executed on the robot. An example of such a plan is given in Fig. 10.3. The plan is purely symbolic and can only handle symbolic concepts. Discrete resource allocation can be considered, but numerical ones cannot. In the example detailed below in Section 10.2.3, the left hand can start the reaching movement as soon as there is enough space in the door opening. A symbolic planner cannot decide the exact date but would issue a condition such as "*the door is open*".

In this work, we assume that the planning phase is provided. We work to its conversion into an executive plan while keeping embedded in the plan the original symbolic plan.

Scheduling: In its general meaning, task scheduling [Baker 09] is the step which comes next to task planning. It consists in determining the adequate timing for each task (start time, completion period and sometimes safety period) to realize the task sequence while fulfilling the constraints of availability of the resources and the temporal constraints. It often aims at minimizing the total duration of the mission, but other objectives can also be considered, such as the minimization of the cost or the respect of the deadlines.

Scheduling provides some numerical data for the task sequence: the starting and ending dates. In robotics, the symbolic map is typically converted into a (explicit or implicit) trajectory, that is to say into purely numerical data. This transformation loses knowledge in terms of semantic: **the original symbolic task plan is lost, which prevents any consultation during the execution**. In fact, this gap [Likhachev 09] appears during the transition between the planning and scheduling phases: whereas the planning phase work with purely symbolic data, the scheduling phase works with purely numerical data, such as geometrical data, temporal data (time constraints, task duration) and resources.

Plan reparation: The execution phase corresponds to the realization of the task sequence using the computed parameters. Because of the unexpected problems or events that may occur during the execution (*e.g.* delays, difference between the expected and real environments, failure of an action), it is most likely that the plan will have to be repaired, *i.e.* adapted in response to situation changes and execution results. Example of a complete planning and repairing framework is given in [Myers 99].

In robotics, where a motion planner is often used to pass from the symbolic plan to the control, a reparation of the symbolic plan would mean to compute again the trajectory, which is costly and poses real questions about the effect of the control of the system (the stable tracking of a changing trajectory may not be a stable control law).

The ambition of the work proposed in this chapter is to provide a methodology to move directly from the symbolic plan to the execution on the robot, without making the trajectory explicit.

10.2.2 Implementation

We consider in the following that a given task planner provides a symbolic sequence of tasks as defined in Chapter 3. A non-linear optimizer is used to compute all the numerical values that are needed to complete the task sequence into a complete controller. This set of numerical parameters is denoted by χ. It typically encompasses the gains of the proportional controllers, the dates of task insertions and removals and the damping factor of the stack-of-tasks solver. The parameter χ defines for each configuration a control law $\dot{q}_\chi$.

Tube of trajectories

Consider an initial configuration of the robot $q(0)$, this control law may be integrated into a unique trajectory $q(t)$:

$$q(t) = \int_0^t \dot{q}_\chi(q(\tau), \Omega(\tau))\ d\tau \tag{10.1}$$

where $\Omega(t)$ generically denotes the configuration of the robot environment at time t. The trajectory is implicit but is completely defined by χ. The task set is a function family spanning a subset of the robot trajectories, for which χ is the parameter family. Of course, if $q(0)$ is changed, or if any unidentified parameter disturbs the integration, another trajectory is generated. Importantly, if the disturbance is small enough, this new trajectory also fulfills all the objectives of the initial map, thanks to the task-function theory: the task sequence in fact defines a tube around the nominal trajectory [Li 08], whose (non-constant) radius is difficult to estimate[2], but which is empirically sufficient to ensure a good robustness at the execution.

Constraints on the tube

Any set of parameters does not systematically define a proper execution. For example, closing the gripper before it reaches the door handler does not allow to further open the door. The controller is local and can converge into some local minimum that makes the execution fails.

[2] as always with the task function

Moreover, all the robot constraints may not be taken into account in the control law. This last consideration is detailed in the next subsection.

All these constraints are checked by a numerical solver. The problem is written with the following generic form:

$$\min_{\chi} f(q, T^{\infty}) \tag{10.2}$$

subject to

$$q(t) = \int_0^t \dot{q}_{\chi}(q(\tau), \Omega(\tau))\ d\tau \tag{10.3}$$

Robot limits
$$\forall t, \quad q(t) \text{ is a correct configuration} \tag{10.4}$$
$$\forall t, \quad \dot{q}_{\chi}(q(t), \Omega(t)) \text{ is a correct control} \tag{10.5}$$

Schedule coherence
$$\forall i, \quad 0 \leq T_i^I < T_i^F \leq T^{\infty} \tag{10.6}$$

Termination condition
$$\forall i, \quad ||e_i(T_i^F)|| \leq \epsilon \tag{10.7}$$

The timings T_i^I and T_i^F configure respectively the insertion and removal of task e_i. The final time T^{∞} ends the execution. All these times are part of the χ variable. The constraints (10.4) and (10.5) check the robot constraints: joint position and velocity limits, obstacle, visibility, etc. The schedule coherence (10.6) checks the obvious constraints that a task should not be removed before its insertion. Causal constraints (such as the gripper should not be closed before the object to grasp is reached) can also be added. The last constraints (10.7) checks the termination of the tasks before they are removed.

A complete description of all the considered constraints is available in [34, 62].

Resolution

The constraints are checked on an explicit representation of the trajectory (10.1). The integration is performed by a mechanical simulator (*e.g.* rigid multi-body [Evrard 08]). The solver is coupled with this simulator. For each evaluated point χ, the trajectory is made explicit by the simulator and the distance to each constraint is computed. The gradient of the cost and constraint functions are evaluated by the solver by finite differences. The solver CFSQP [Lawrence 97] was used for both the resolution and the gradient estimations.

The above problem can roughly be directly converted into a SQP for numerical resolution. Attention has to be paid to the formulation of semi-infinite problem: it includes a finite number of parameters χ, but has infinitely many constraints such as (10.4) and (10.5). A method to handle such constraints is to set one constraint on each time sample of the integration grid. However, this leads to two problems: it adds a high number of constraints and this number varies with T^{∞}. If considering a rougher grid, the number of constraints diminishes but multiple local maxima are likely to appear on a single grid sample, which causes continuity problems. Alternative tricks can be used to provide a better solver behavior with a reduced resolution cost. They are discussed in details in [62].

Constraints in the planner or in the control

This subsection is a short but digressive discussion about the level of definition to take constraints into account. Robot constraints, such as typically the collision avoidance, can be taken here into account by two means: by a constraint in the controller ($\dot{q}_{\chi}$ would always

avoid the obstacle) or only in the non-linear sequence optimizer. Similarly, in sampled-based path planning [Barraquand 92, Kuffner 00] (typically rapidly-exploring random tree – RRT), the obstacle can be taken into account at the level of the local steering method or only by the collision checker that builds the graph. In any case, the graph would be collision free, but in the first case, many more connections would be enabled for fewer nodes. In other words, the *visibility* (*i.e.* the volume of the neighborhood around each node that is reachable by the local method) of each node is enlarged.

For both SQP and RRT, taking into account the obstacles in the local method reduces the number of steps of the overall planning method is reduced but increases the cost of each step. In [Dalibard 09], a rigorous experimental analysis was proposed to compare the cost of a single obstacle-aware local method (no random iteration, the goal is visible from the starting point with the considered local method) to a classical RRT propagation whose local method does not take care of the obstacles. The first only needs a single iteration to converge but was in total more costly than the iterative RRT.

Similarly, here, the obstacles may be taken into account into the control law that is integrated by the simulator. The total cost is increased, since it is more efficient to consider the obstacle at the SQP level than at each step of the integrated QP control law. However, the meaning is not the same: when the SQP checks the constraints alone, the resulting trajectory avoids the obstacles "*by chance*": a little disturbance on the process leads the trajectory into collision, since nothing in the control law prevent it from occurring. In other words, the tube around the nominal trajectory is improperly defined when going around an obstacle.

In the following experiment, the constraints were not taken into account by the control law but only by the solver. The other solution is possible. To avoid any supplementary costs, this integration should be carefully performed to avoid a double check by both the QP (control) and SQP (sequencer). In any case, the controller alone has to take care of all the details during the real execution on the robot.

10.2.3 An example: grasp the can out of the fridge

The sequence of tasks given in Fig. 10.4 describes a robot taking out a can from the fridge. The corresponding tasks are:

- e_0: Open the right gripper
- e_1: Move the right arm to the fridge
- e_2: Close the right gripper
- e_3: Open the fridge
- e_4: Close the fridge
- e_5: Open the left gripper
- e_6: Move the left gripper in the fridge area
- e_7: Move the left gripper to the can
- e_8: Close the left gripper

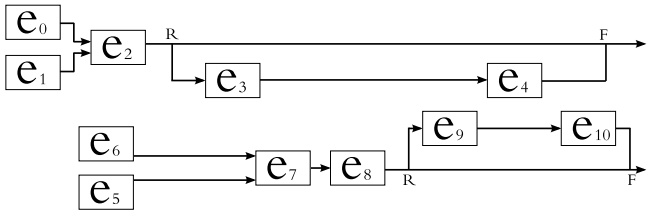
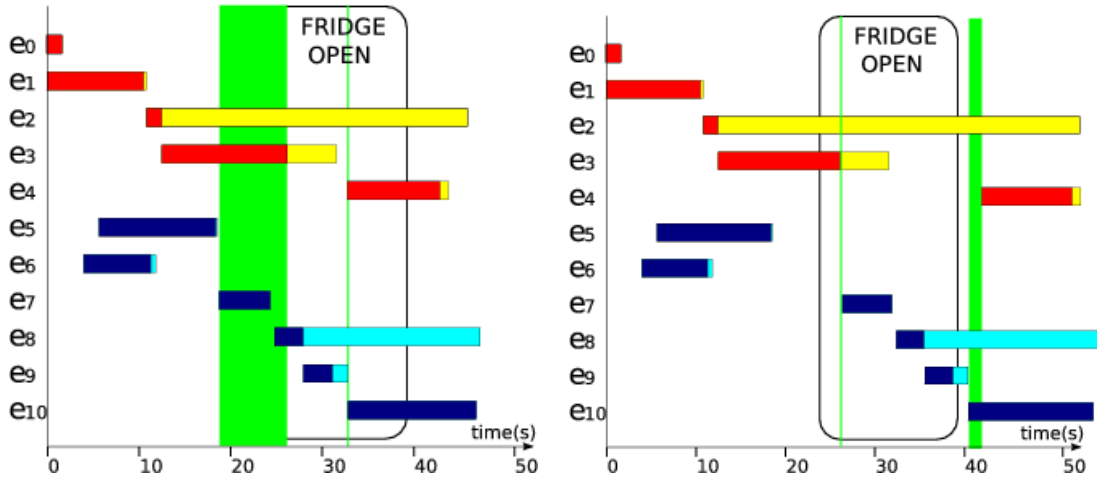


Figure 10.4 *– Sequence of tasks to be executed by the robot. The arrows represent the causal links between tasks. An "R" is added when a the next task should wait for the regulation (error nullified) of the previous task to be inserted while keeping the previous task active. Both threads then join with an "F" that removes the joining task.*



(a) Optimal (b) Sub optimal

Figure 10.5 *– Summary of the execution schedules. The left-hand tasks are in red and yellow (red when the error is not zero, yellow when the task is regulated). The right-hand tasks are in blue (dark blue when the error is not zero, light blue when the task is regulated).*
The optimum takes advantage of the bi-manual robot configuration and launches the left-arm reaching task while the fridge is not fully open. The sub-optimal waits for the fridge to be completely open.

- e_9: Lift the can
- e_{10}: Remove the can out of the fridge

In particular, no causal links are given between the end of e_3 (fridge is open) and the beginning of e_6 (right hand moves). This link is implicit and is computed by the collision checker inside the simulation. The cost function is simply the total time of the mission T^∞.

The comparison between the schedule with and without optimization is given in Fig. 10.5. The first sequence is the result output by the solver for the graph Fig. 10.4. The second sequence is the result when an explicit causal link is added between the right e_3 and left e_6 subgroups. This second optimization corresponds to the best that can be expected when accounting only for symbols. In the first case, the total time is $T^\infty = 44s$ while it is $T^\infty = 56s$ in the second case. With a dummy settings χ (medium gains and non-overlapping tasks), the

Optimization of tasks warping and scheduling for smooth sequencing of robotic actions

F. Keith, N. Mansard, S. Miossec, A. Kheddar

IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09) [34]

Context:
This paper was written during F. Keith thesis. The scientific project behind the thesis is still open and is now part of the FP7 Robohow.cog project.

Motivations:
The stack of tasks can be seen as a converter between a symbolic plan and a numerical execution. However, the tasks are not pure symbolic objects, but rather proto-symbols that contains a part of numerical data (e.g. gain, time of start and end). The objective of this work is to give a complete solution to pass from a symbolic sequence of tasks to the execution by the robot.

Approach:
The method proposes to compute all the numerical data to complete a sequence of tasks given by a symbolic planner. A non-linear optimizer is coupled to a robotics simulator: at each trial, the solver chooses a set of numerical data and the corresponding sequence is executed in simulation. The distance to the constraints (joint position and velocity, obstacle, etc) is computed by the simulator and feedback to the solver. The solver then iteratively converges to the optimal parameter values with respect to the given cost, for example minimum execution time.

Results and contributions:
The method effectively solves the problem and manages to find the best execution parameters that satisfies the sequence and robot constraints. It was used to optimize a classical robotics example: *grasp the can from the fridge*.

Limitations and perspectives:
The problem is highly non-linear. Moreover, the gradients of the problem seems impossible to analytically compute. The convergence time is then very slow and the solver is often stuck into some local minimum. There is still some work to do on the problem formulation.
During the execution by the robot, the parameters are fixed and, considering the computation cost, cannot be re-optimized in real time in case of changes of the environment. The gradient issued from the solver could be used to close the loop on the sequence parameters.

total time was 71s.

The motion was then executed by the robot with an impedance force-feedback task to drive the right hand while opening the fridge. Snapshots of the execution can be found on page 98.

10.2.4 Discussion and limits

The developments proposed in this chapter are only a first step in the direction of using the task function to bind reasoning and execution. In practice, the proposed method suffers of one big limitation: its cost. The solver needs minutes of computation to compute seconds of motion. There is a lot to do to optimize the solver. In particular, the simulation capabilities are a key feature of the robot reasoning, while here a basic constant-timestep integrator is used. We can hope to save a lot of time easily.

The biggest limitation comes from the high non-linear couplings between the parameters and the constraints. Like many indirect systems, the control at the beginning of the sequence impacts the trajectory at the end of the trajectory. Moreover, it seems difficult to compute the corresponding gradients analytically. Consequently, the solver difficultly estimates the gradient, while lack of precision in the finite difference leads to improper descent direction. Moreover, the complexity of the resolution increases with the length of the mission, even if some sub parts may be nearly decoupled from each other.

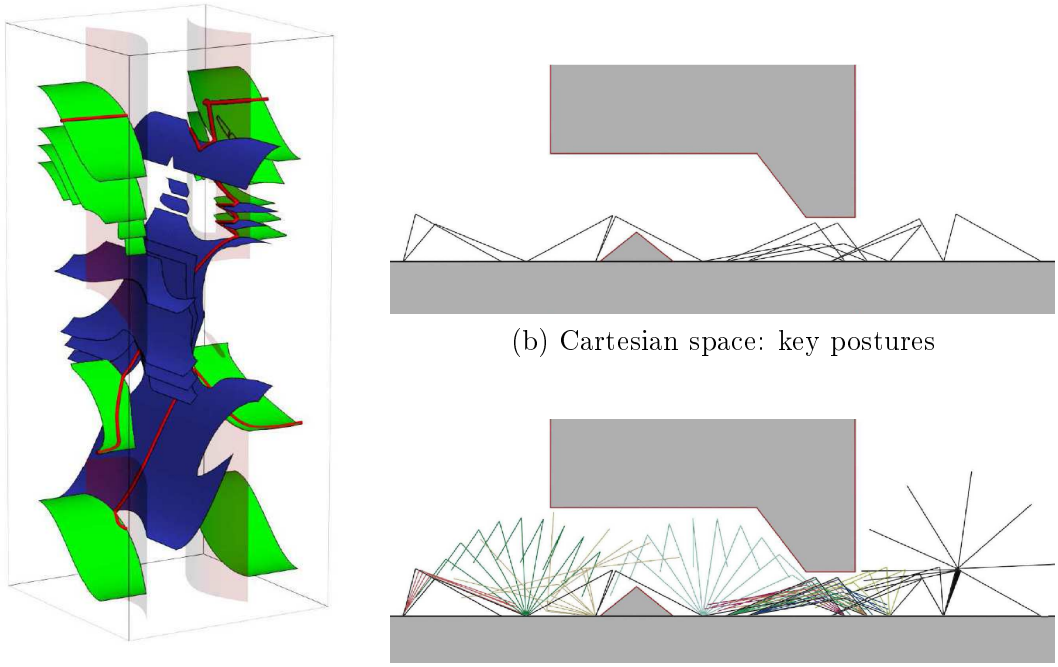
10.3 Towards task-based motion planning?

In the previous section, we assumed the original symbolic sequence to be given by a symbolic planner that has few numerical knowledge about the world. The sequence optimizer then takes the position that is commonly occupied by a sampled-based motion planner, that is to say to handle all the geometrical and numerical aspects that the symbolic planner cannot manipulate.

Such a sampled-based planner could also work with task-based trajectories. Indeed, sampled-based planners are based on local steering method that drives the system from one node of the graph to the other. This local controller is typically based on an objective written in the configuration space for simple manipulators targeting a desired configuration. For systems that are redundant with respect to the target (typically, a 7 DOF arm targeting a position of the end effector) or that are subject to additional constraints (typically, the humanoid robot that should keep contact with the ground), the local controller cannot be written by a simple objective in the configuration space.

This aspect is explored in humanoid robotics but is formerly known as the manipulation planning problem [Siméon 04]. The similarity with the structure of the sampled space when considering humanoid (or generally legged) robots was done in [Hauser 08]. Consider first the problem of contact planning for humanoid robots: among all the possible configurations, only those in statically-stable contact are acceptable. The configuration should be sampled in submanifold of the configuration space [Escande 08b, Mordatch 12]. The best understanding of the resulting folded structure can be found from the theses of A. Escande [Escande 08a][3] and K. Bouyarmane [Bouyarmane 11a]. An example for a simple two-bodies robot taken from

[3] alas in French with a summary in English apart in [Escande 13]



(a) Configuration space

(b) Cartesian space: key postures

(c) Cartesian space: complete trajectories

Figure 10.6 *– (a) Layer structure of the configuration space (b) Cartesian space key postures (c) chronophography of the interpolated robot motion. Images from [Escande 08a]*

[Escande 08a] is given in Fig. 10.6. The configuration space is composed of multiple layers that correspond to sets of contacts. The robot travels from one layer to the other with a quasi-symbolic decision to add or relax one contact point.

The interesting aspect of contact planning is that all contacts are equivalent in a functional point of view. A similar structure is highlighted in [Siméon 04] for manipulation. Some semantics has then to be attached to the description of the environment to make explicit that all the submanifolds are not equivalent in their function. A generic framework to include such a semantic is developed under the name *Documented Objects* [Dalibard 10]: each relevant object of the map is augmented with a description of its affordance, given as the sequence of tasks to activate it. Such an approach melts both symbolic and geometrical search. It issues a task sequence that can be nearly directly used as the working controller on board the real robot to execute the plan. In that work chain, the sequencer optimizer would be an interesting feature to smooth the random trajectory output by the planner. Many problems arise when considering a realistic implementation. It constitutes one of the main perspectives developed in the final chapter.

Chapter 11

Motion description

In the previous chapter, we have seen that the task-function approach is an appealing solution to describe a motion to be executed by the robot and that it is possible to reason with this representation. In this chapter, it is shown that the same approach can be used to describe an observed motion. Of course, if a task-based description is extracted during the recognition, it is straightforward to replicate it on any similar-shaped structure for imitation. In a first time, we show the interest of the approach in recognizing ambiguous motions performed by the HRP-2 robot in Section 11.2. We are now working to fuse this approach to the possible models of human body and action, as described in Section 11.3.

11.1 Introductory example

Reference paper [19]

The *Standing Lotus* yoga motion was an internal challenge in our team, to design and run on the robot a given dynamic motion (standing on one leg with an expressive posture) using the motion-capture system and the motion-generation tools described in the first part, in a single day of work. The initial motion was demonstrated by L. Saab on the morning and recorder by the motion capture system (Fig. 11.1-(a)). The marker trajectories were targeted to the robot geometry using, for each time step, a inverse geometry minimizing the distances of each body of the robot to the associated body of the human teacher (Fig. 11.1-(b)). Two problems clearly appear: the hands collides (Fig. 11.1-(b)-v) and the COM is not properly positioned, which makes the robot falls while the human strikes balance (Fig. 11.1-(c)). If trying to directly execute the motion into a dynamic simulator, the robot falls since the COM is not below the support foot when reaching the still pose.

The dynamically-consistent motion is finally obtain with the addition of one task to regulate the placements of the hands and remove the auto-collision and one task to track the COM. The result is stable, collision safe (Fig. 11.1-(d)) and can be replayed by the robot

(Fig. 11.1-(e)). This semi-automatic motion targeting from human-motion capture is the basis of the *"Dance with HRP-2"* demonstration, presented in the next chapter.

The conclusion of this example is that the quality of the imitation depends on the choice of the variables that are relevant for the motion context. For the yoga motion, the balance is the first purpose of the motion. Then the position of the hands in front of the chest is considered. The global posture only comes last. If asking a yoga teacher, other relevant variables would certainly be proposed to improve the rendering of the motion. To automatize the imitation process, one key point is to automatically detect which are the relevant variables. This is the purpose of the next section.

11.2 Task recognition

Reference paper [9] (see also [47, 27])

11.2.1 Overview and hypothesis

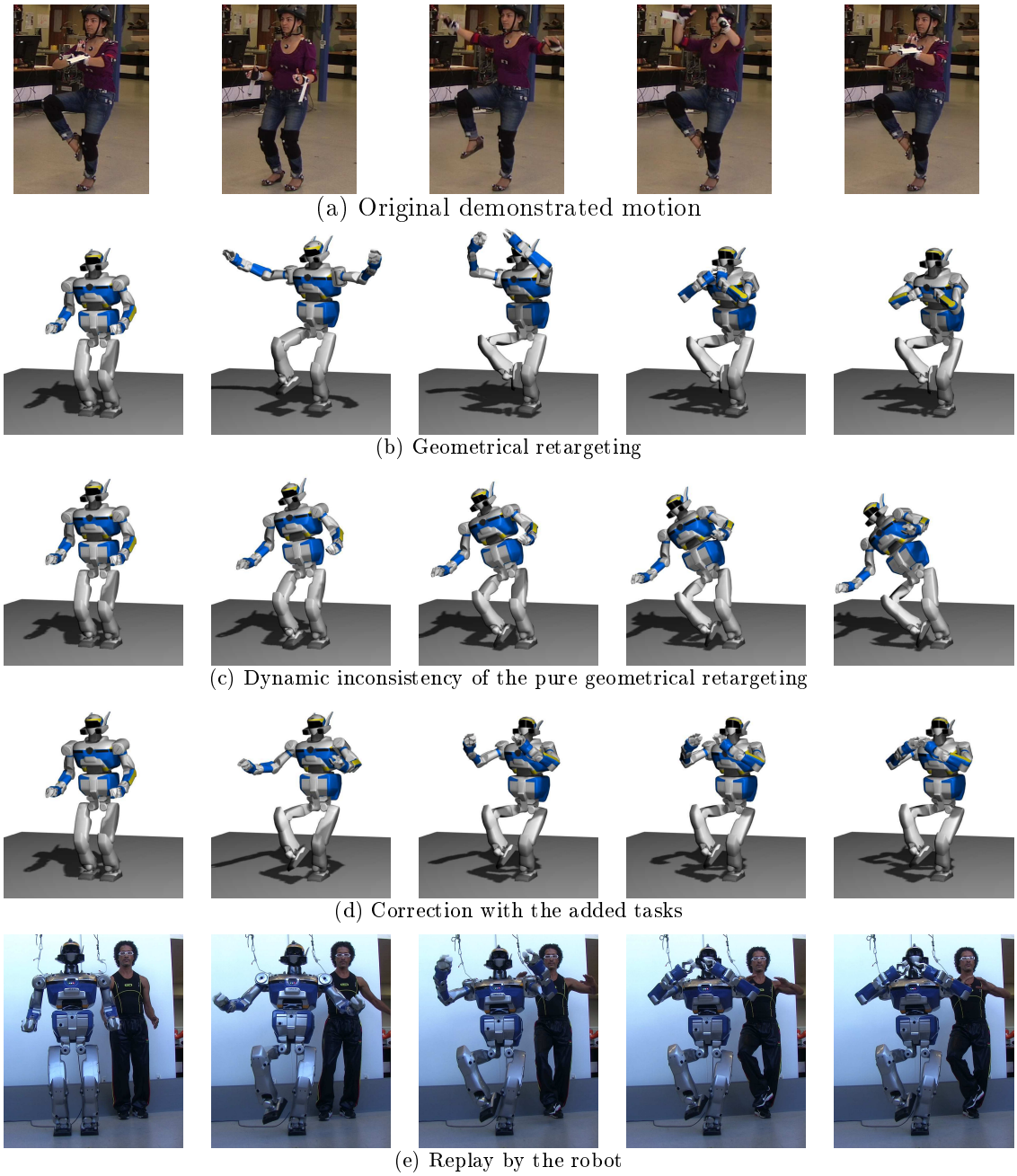
In a first time, we consider the recognition of a motion performed by the robot, where all the possible controllers are known. We suppose that the considered motion has been generated by a stack of tasks and that all the possible tasks are known. The observed motion is given by a trajectory $t \in [0, T] \rightarrow \dot{q}(t)$. The selected stack is supposed to be constant during the whole motion (no task insertion or removal). The recognition then consists in selecting in the pool of possible tasks the active ones. Like in the previous chapter, a task is then seen as a symbol that categorizes the movement. However, a task contains some numerical quantities that make it a proto-symbol. The task recognition simultaneously selects the active tasks and computes the numerical data (position of the reference values, gain, etc.). For the experiments presented below, the joint velocities are reconstructed from the motion-capture data using non-linear optimization. No other contextual information (such as prominent objects of the environment) or repetition of the motion is used to recognize the motion.

11.2.2 Iterative task selection

The selection algorithm iteratively chooses the most relevant task from the pool, by computing the optimal numerical values fitting the task on $\dot{q}(t)$. For the next iteration, the algorithm then nullifies the effect of the task to avoid future false detection. The algorithm loops until the initial velocity $\dot{q}(t)$ is completely canceled by successive task removal. The algorithm is summarized in Algorithm 1. The two main parts (fitting and projection) are described below.

Task fitting

This part of the algorithm does the selection of the most relevant task. Each task is defined by its task function $e(q)$ and a vector field in the task space $\dot{e}^*$ (following the definition given in Chapter 3). Consequently, if a task is active during the motion generation, the image of the trajectory $\dot{q}(t)$ in the task space should fit with the vector field. This is characterized by the distance of the tangent to the trajectory in the task space to the vector field. Since the vector field is depending on some numerical parameters of the proto-symbolic task denoted



(a) Original demonstrated motion

(b) Geometrical retargeting

(c) Dynamic inconsistency of the pure geometrical retargeting

(d) Correction with the added tasks

(e) Replay by the robot

Figure 11.1 *– Standing-Lotus robot challenge: from human demonstration to dynamic robot replay.*
(a) The initial demonstrated movement: the trajectory of each marker is recorded by the motion-capture system. (b) Each body of the robot is positioned to minimize the distance with the associated body of the human. (c) If replaying the geometrical extracted motion into a dynamic simulator, the robot falls: the COM of the robot is not properly positioned to ensure balance. (d) Retargeting of the hands and COM positions in priority, posture second: dynamic replay, the robot is balancing. (e) Replay by the physical robot with, in real-time, the dancer T. Benamara imitating the robot.

Algorithm 1 Task selection

Input: trajectory: $t \in [0,T] \rightarrow \dot{q}(t)$
task pool: e_1 ... e_N
Output: k selected tasks e_{s_1} ... e_{s_k}
associated numerical data $\chi_{s_1} ... \chi_{s_k}$
$v(t) := \dot{q}(t)$
repeat
  for each unselected task i **do**
    ## – – Compute numerical data χ_i and residue r_i – –##
    $\chi_i, r_i :=$ taskFitting$(i, v(t))$
  end foreach
  ## – – Select the task with minimum residue – –##
  $s :=$ argmin r_i
  Select task s
  ## – – Nullify the task effect – –##
  $v(t) := P_s v(t)$
until $\int_0^T ||v(t)||dt < \epsilon$

χ, we choose the parameters that enable the best fitting:

$$\chi^* = \min_{\chi} . \int_0^T ||J(t)\dot{q}(t) - \dot{e}^*_\chi||^2 dt \tag{11.1}$$

In general, this is a non-linear minimization problem that is solved by a SQP. The residue r^* corresponding to the optimum χ^* can be used as a distance that characterizes the given motion $\dot{q}(t)$ the model of the task. Among all the tasks of the pool, the one that has the smallest residue is the most relevant to be active. To avoid any scaling problem, the residue is normalized. See [9] for details.

An example of a discrimination between two candidate tasks is given in Fig. 11.3.

To select a set of tasks that explains the observed motion, a solution would be to take all the tasks whose fitting residue is below a given threshold. However, the threshold is difficult to set up and many false alarms may arise. If comparing the two graphs of Fig. 11.3, the right one follows indeed the task model, but the left one is not so far neither. This is because the motion in the left space is the image of the motion in the right space and thus looks similar with a transformation that might be nearly linear.

Indeed, some tasks of the pool are linked together by a nearly-linear relation. The trajectories in these neighbor tasks are then closely related. For example, if a task driving the tip of the right arm effector is active (*e.g.* the tip follows a straight line), then the right wrist will have a very similar trajectory and the elbow would also behave similarly. Another example: if the COM moves from one foot to the other and the chest is straight, then the point between the shoulders would move from one foot to the other. This effect is emphasized by biomechanical studies [Latash 02]: if studying the repetition of a same motion by a human subject, the ratio between the variance in the task space and in its null-space (called uncontrolled manifold) decreases when coming closer to the controlled point.

Projection

To avoid the false alarms due to linked tasks, the effect in the observed motion of the detected task is canceled. This is simply obtained by projecting the observed joint velocity in the null space of the detected tasks. The following task will be detected from this projected velocity:

$$\dot{q}(t) := P(t)\dot{q}(t) \tag{11.2}$$

The projector P depends on the robot configuration and is computed from the observed non-projected configuration.

It is possible to prove first that, for a perfect observation, the successively projected joint velocity becomes zero when and only when all the active tasks have been detected; and second that the order of detection of the tasks does not influence any further detection (see [9] and [64] for details).

11.2.3 Results

We have validated the approach with motions demonstrated by the HRP-2 robot. The motion is performed by the robot and recorded with the motion-capture system. The robot configuration is reconstructed by non-linear optimization of the marker models.

The algorithm is very accurate, even in noisy conditions. It is used to disambiguate similar-looking motions. Consider the example of Fig. 11.2: two similarly-looking motions are compared. On the first sequence (Fig. 11.2-(a)), the robot bends forward to reach a ball while keeping a fixed COM. Its left hand automatically moves to correct the balance. On the second motion (Fig. 11.2-(b)), both robot hands move to a target position: the right hand moves toward the same target than in the first motion; the left hand moves toward the position reached in Fig. 11.2-(a). The COM is also controlled at a fixed position. The purposes of the two motions are different: the left hand is free in the first case and is controlled in the second. Both motions look very similar to the human eye.

The results of the fitting for the left hand are shown in Fig. 11.3. During the first movement, the left hand moves to correct the balance. Its trajectory does not follow the task model, as shown Fig. 11.3-(a). During the second motion, the left hand is controlled to a given target and its trajectory fits the task model very well, as shown in Fig. 11.3-(b).

The algorithm stops when all the motion can be explained by a task. The successive projections for both motions are displayed in Fig. 11.4. The first motion needs three projections to be completely canceled (right hand, COM and feet tasks). The second motion needs four projections (right and left hands, COM and feet tasks).

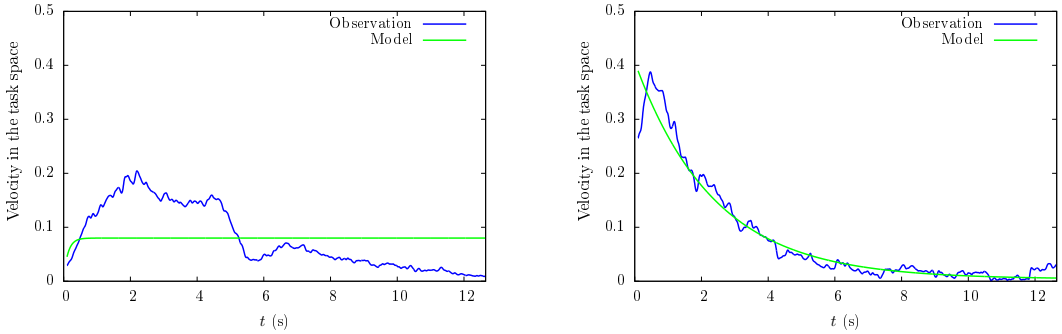
11.3 Human-like motion generation

Reference paper [7] (see also [63, 64])

The interest of this approach is to formulated the recognition as a dual problem to motion generation. If we are able to generate some movements for a system, we are also able to recognize and reconstruct the generation method from the observation only. The question is then to know if it is possible to generate plausible motions on human avatars. Considering the chosen task-based approach, the question is divided in two points:

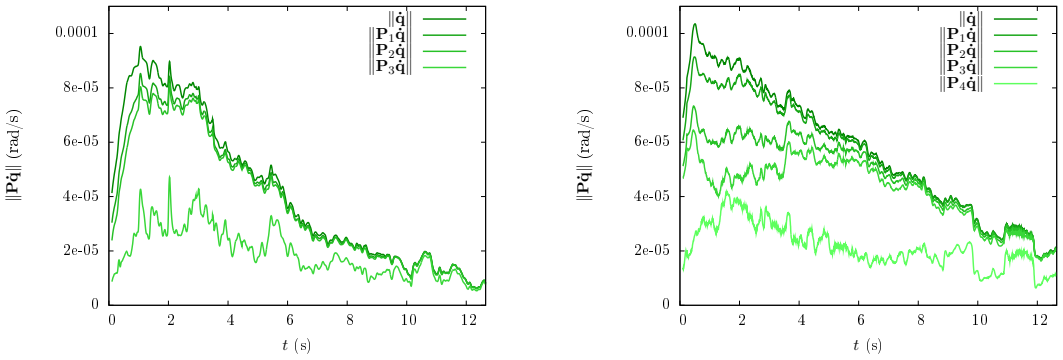
(a) First motion (b) Second motion

Figure 11.2 *– Final pose for two similar-looking motions. (a) Right grasp, the left hand moves to regulate the robot balance. (b) Right and left grasps, the left hand moves to the final position reached during the first motion.*



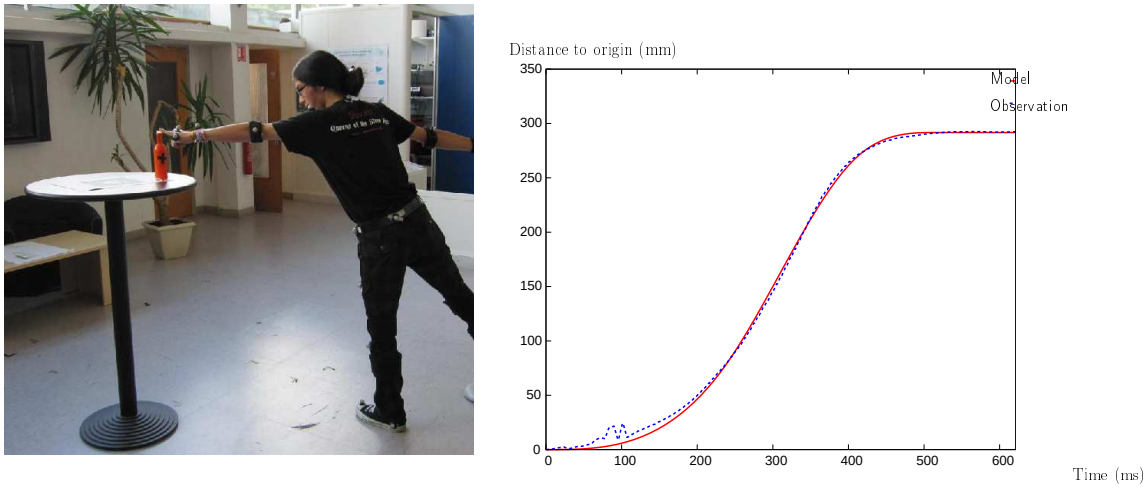
(a) Negative fit (b) Positive fit

Figure 11.3 *– Two examples of the optimized best fit of a task model (proportional $\dot{e}^* = -\lambda e$). (a) First motion, uncontrolled left hand: the model does not fit well, the task is likely inactive. (b) Second motion, left hand controlled: the model fits well, the task is a good candidate.*



(a) First motion (b) Second motion

Figure 11.4 *– Decrease of the quantity of movement $||\dot{q}(t)||$ after each projection of the detection loop. (a) First motion: three tasks are active, the quantity of motion is zero (plus noise) after the three projections. (b) Second motion: four tasks are active, there is residual motion after the effects of third task are canceled. The projection into the null space of the left hand finally nullified the quantity of motion.*

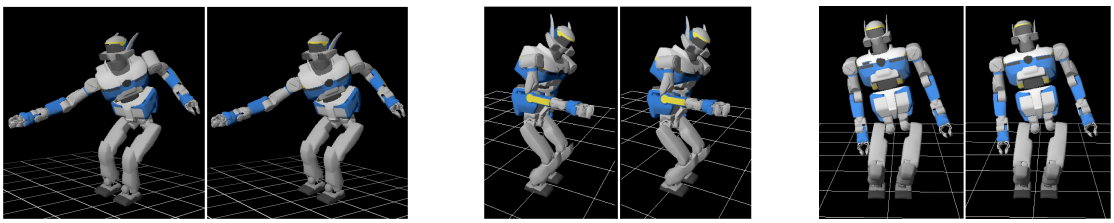


(a) Reaching while balancing (b)

Figure 11.5 *– Example of minimum-jerk trajectories. (a) The human is asked to reach a distant object with the left fingertip, while maintaining his balance on one foot and keeping his right hand fixed. (b) The trajectory of the left hand follows a minimum-jerk rule, which is easily recognized by model fitting.*

Reverse control for humanoid robot task recognition

S. Hak, N. Mansard, O. Stasse, J-P. Laumond



IEEE Transaction on System, Man and Cybernetics [9]

Context:
This is paper was written during S. Hak thesis in the R-Blink project. The ultimate goal is to recognize the on-going task to give after very small computation time ("in the blink of an eye") an appropriate answer or an instantaneous imitation.

Motivations:
When replicating a motion, several spaces can be chosen to compute the distance to the original motion. However, in a given context, only one of them is relevant. The idea of the paper is to automatically recognize the relevant space, which would then be used as the reference task space to replicate.

Approach:
The task recognition is performed only from the body trajectories without using any contextual information: the input is the configuration trajectory of the demonstrator. The motion is then projected in each task space of a candidate task pool and fit with a reference trajectory. The residue of the fitting is used to select the best task among the candidates. The effect of this task is then canceled from the original motion by projecting it in the space orthogonal to the task. The iterative selection finally ends when the residual motion is null.

Results and contributions:
The approach was applied to disambiguate similar-looking motion of HRP-2. A real experiment was achieved using the motion capture system to realistically acquire the robot motion. The method produces very good performances even in the presence of noise.

Limitations and perspectives:
The set of possible tasks has to be known beforehand. This is a strong assumption when working with human. A preliminary experiment was set to recognize minimum-jerk trajectories of the human hand during a pointing task, but the generalization to any human motion is still an open topic.

- Is there some characteristic trajectories in a space linked to the executed action?
- what are the combination rules to generated the whole-body trajectories from several given objectives?

11.3.1 Model of task

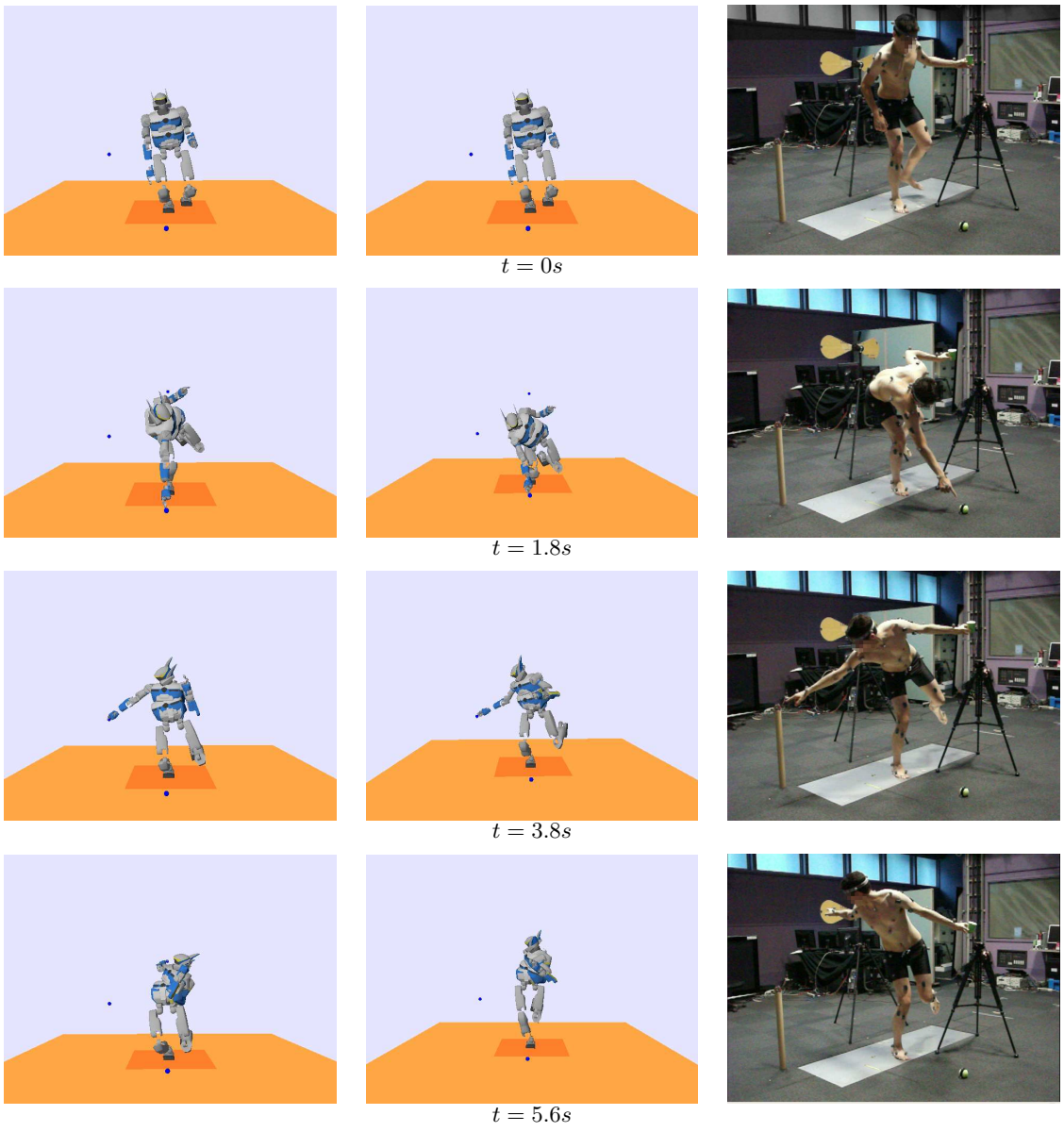
The task-based approach, consisting in characterizing the on-going action into a reduced space, seems relevant in view of some biomechanics studies [Latash 02, JacquierBret 09]: for a repetition of movements, the variance is lower in some spaces intuitively linked to the objective of the motion while the variance increases in the orthogonal to this space. However, the trajectory itself may not be a characteristic of the observed movement. For example, we have considered in a preliminary experiment the motion of the tip of the finger in a distant-reach task (see Fig. 11.5-(a)). The trajectory is a classical minimum jerk, easily detectable with the fitting optimization problem formulated in the previous section (see Fig. 11.5-(b)). However, this trajectory may not be characteristic, as we can expect to get a similar answer from other parts of the body.

11.3.2 Task composition and motion generation

From a human demonstration, we can extract some features that are intuitively linked to the demanded objectives: the hand motion when reaching, the COM or ZMP trajectories if balancing matters, etc. From these basic features, we now would like to reconstruct the motion of the whole body, *i.e.* the trajectories of all the joints angles and torques or the activities of the muscles. Generation of realistic human motion is an objective *per se*, that goes far beyond the work presented in this chapter. The preliminary work presented below makes the link to vaster perspectives that are developed in the final chapter.

This objective is partially addressed in many topical works, where the muscle activity is reconstructed from an accurate tracking of all the parts of the body, with if possible additional data such as the evaluation of physiological states of muscles with electromyography or the contact application forces with force plates [Rasmussen 01, Nakamura 05]. Here, we take the hypothesis that all the possible parameters of the body model should be taken into account and participate to the naturalness of the generated motion. A first step in this direction is to use the inverse dynamics presented in Chapter 6 to solve at the same time the kinematics and the dynamics of the system. Among the possible redundancy left for a given motion, we are expected that the dynamic information is leading to choose a motion that looks more natural. Such approach often relies finally on the choice of the most appropriate cost function that can be used for the optimization [Chadwick 09]. The dynamics of the system by itself acts as a cost function that drives the choice among the redundant movements.

Fig. 11.6 shows a preliminary experiment. As previously, the human is asked to reach an object with the fingertip (to avoid any problem due to the orientation of the hand) while standing on one foot (to avoid any problem due to a close kinematic chain). The same objectives are given to an avatar whose inertia parameters are those of the human subject and solved once by inverse kinematics and once by inverse dynamics. Motion artifacts appear in the motion obtained by inverse kinematics: the left arm drifts upward, the chest unnaturally tilts backward, the neck takes a strange angle. Those artifacts disappear when generating the motion by inverse dynamics. Differences remain with the human performance that may be



(a) Inverse kinematics (b) Inverse dynamics (c) Human performance

Figure 11.6 *– Comparison of a motion generated by inverse kinematics or dynamics with a real human performance. The avatar (a) and (b) has the same characteristics as the human subject (dimension and inertia) although the appearance is based on HRP-2 (for technical reasons). Motion artifacts appear in inverse kinematics that are naturally canceled when taking into account the body dynamics.*

removed if taking a more accurate avatar model (taking into account the muscle actuation redundancy for example). Similarly, the joint torques generated by inverse kinematics and dynamics where compared with the torques reconstructed by inverse dynamics of the human performance. As expected, the torques trajectories are closer to the natural ones using the inverse dynamics.

11.4 Conclusion

The main contribution of this part is to write the trajectory recognition problem as a dual problem of the motion generation problem. The same paradigms can thus be used in direct (generation) and inverse (recognition) motion problems. The chapter opens many perspectives both in motion recognition and imitation and in generation of human-like motions and human model animation. The problem was here introduced in the context of the motion recognition, but it originally comes from biomechanics for the understanding of human behavior models. Many applications directly comes from this field, in computer animation (for realistic rendering and digital mock-up evaluation) or in robotics (for human-robot interaction). Animation of complex-actuation musculoskeletal models also makes a link with recent mechatronics development, where co-contraction (variable stiffness) plays a central role. These perspectives are developed in the last chapter.

Chapter 12

Demonstration

This chapter aims at giving an integrative overview of the methodologies developed in the document, by presenting two demonstrators based on the stack-of-tasks approach and software for the humanoid robot HRP-2. The first demonstration is an applicative presentation the humanoid robotics as a tool for collaborative working environments (CWE). It is an overview of the capabilities of the humanoids for navigating and executing manipulation tasks while standing or walking, in remote control or in autonomy, but in any case when low cognitive charge is expected. The second demonstration emphasizes HRP-2 dancing in open-loop during a live event. It is a demonstrator of the capabilities of our motion generation methods. The realization of these demonstrators was the opportunity to validate the capabilities of our approach seen as a robot motion programing language.

12.1 Robot at collaborative working environments

Reference paper [3] (see also [35, 33, 30])

"Robot@CWE" was the opportunity to demonstrate a complete integration of the actions that can be expected from a humanoid robot working in a CWE: autonomy for simple tasks like navigation in a known environment, passive physical cooperation to help a local operator and teleoperation for the fine-manipulation tasks. The scenario places the HRP-2 robot in a construction site in Japan, working with both local operators physically present on the site, and specialist distant operators working by teleoperation through the robot. The integration was led by O. Stasse and the finalization of the demonstrator was performed at JRL-Japan by O. Stasse and P. Evrard in collaboration with the FP7 Robot@CWE consortium.

The global sequence of the demonstration is given in Fig. 12.1. The overall objective "*assemble a heavy object on the construction site with a local operator*" is build by a distant manager inside a virtual world. The robot autonomously navigates in the construction site to position itself in front of the object to assemble. The precise positioning of the robot effector is performed in teleoperation from Germany. The teleoperation then stops and the robot

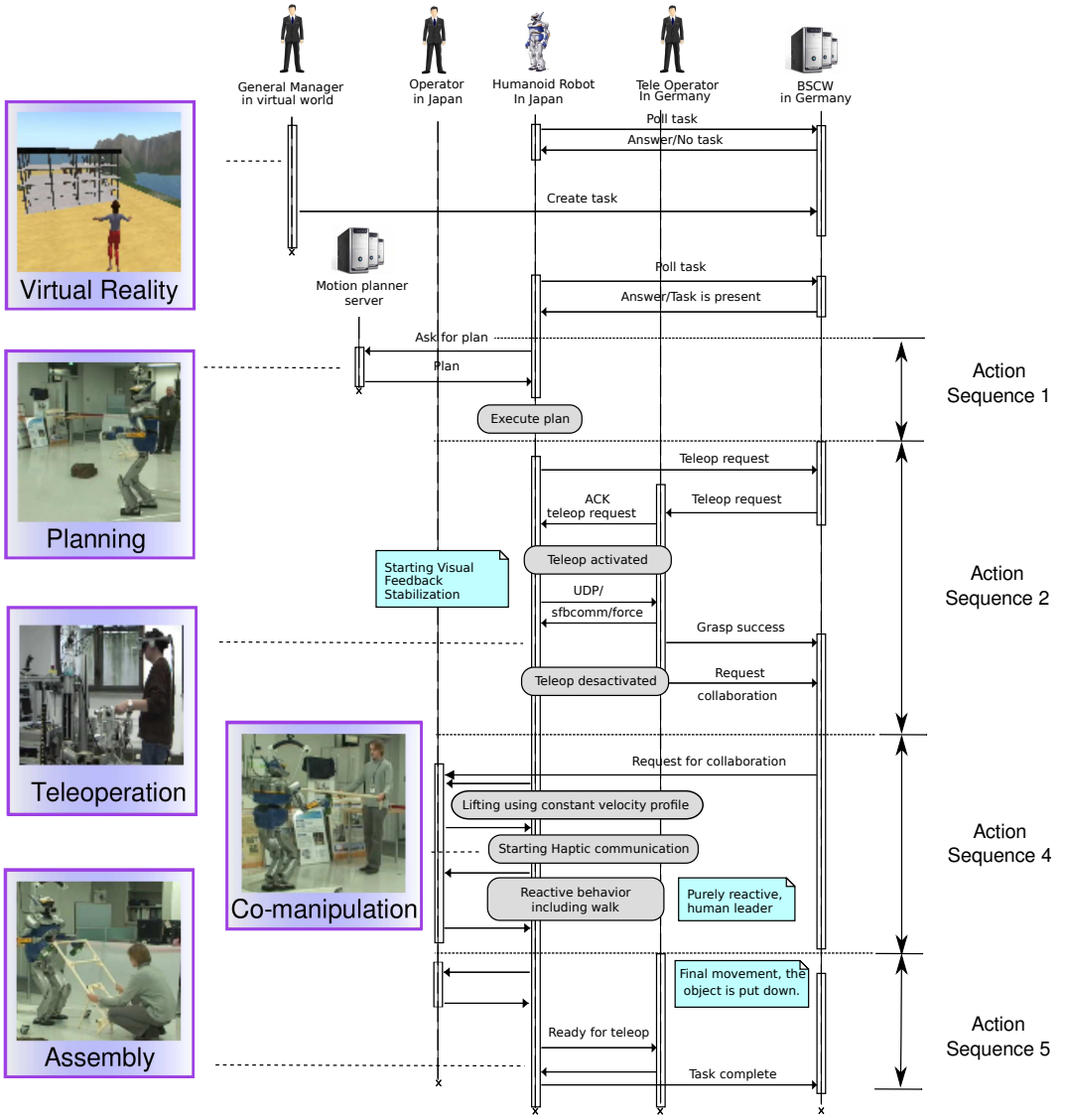


Figure 12.1 *– Robot@CWE: UML sequence diagram.*

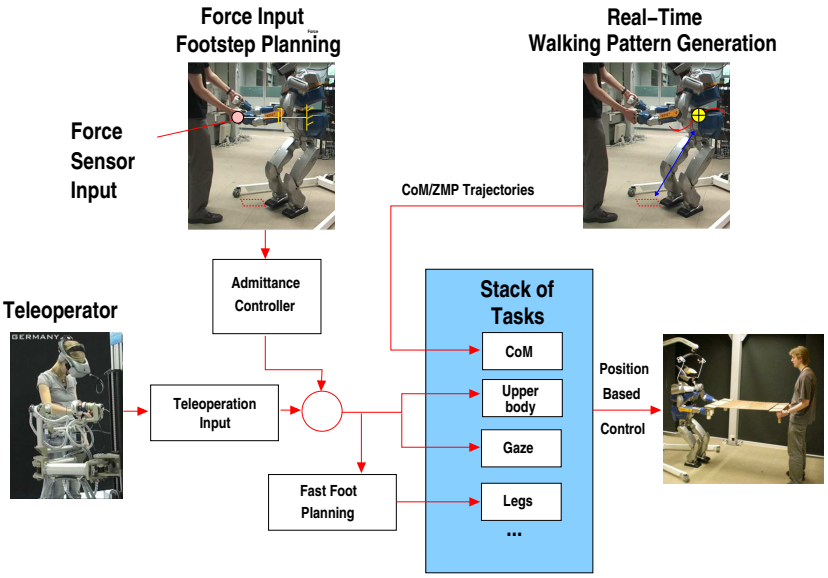


Figure 12.2 – *Robot@CWE: Robot control kernel.*

starts a physical collaboration with a local operator. In a first time, the robot is follower and the operator guides it to the assembly spot. In a second time, the robot takes the lead and performs the fine assembly task with the assistance of the human.

The core of the robot motion generation is summarized in Fig. 12.2. The control is composed of active-compliance (impedance) tasks to move the hands to which reference forces are fed in during teleoperation, control of the gaze orientation from the references sent by the distant operator. The COM trajectories are computed from the preview control based on a linear inverse pendulum model [Morisawa 07]. The flying foot is modified using a fast feasibility test [Perrin 10]. All the tasks are composed together using the classical stack of tasks (3.26) presented in Part I and integrated using the subsequent software. All the details about the robot sequence and the control implementation are given in [3].

The demonstrator proved to be very robust. It was validated with ten users that did not know anything of robotics, acting as the local operator on the construction site. The robot functioning model (switch between teleoperation and autonomy, physical cooperation, etc) appeared clear to the users. The central part of the demonstration (carrying an object in cooperation with the walking robot) was also used as a standard demonstrator of the laboratory and was tested by a hundred of visitors.

This demonstrator proves first that our software and the surrounding methodology enable the development of composed robotic movements inside a complex applicative framework. Thanks to the use of sensor feedback, it is possible to obtain a good robustness. Finally the demonstrator shows the practical interest of the humanoid robot for an industrial process and gives an idea of the maturity of this technology, with one key missing point: the robustness of the walk and of the robot balance in general.

Multi-modal collaborative work with humanoid robots: a case study with hrp-2

O. Stasse, P. Evrard, N. Mansard et al.

To be submitted soon [3]

Context:
This paper synthesizes the work done around the demonstrator of the FP7 *Robot @ Collaborative Working Environment* project, led by A. Kheddar.

Motivations:
The ambition of the Robot@CWE project are to show some case studies where a humanoid robot is used in a working environment for autonomous, teleoperated, single or cooperative tasks.

Approach:
The scenario shows the HRP-2 robot autonomously reaching a location after a request of a distant user for a surveillance task. When in front of the object, the robot is driven by a long-distance teleoperator to grasp an object. It then autonomously carries the object in collaboration with a local operator, bringing the object some step away. A cooperative positioning task is finally performed by the robot and the local operator.

Results and contributions:
The paper shows the first real implementation of a sequence of cooperative tasks performed autonomously or in teleoperation by a humanoid robot. Moreover, a complete case-study with non-specialists local operator was performed, that proofs the robustness of the demonstrator and the interest of the humanoid robot in realistic scenarios.

Limitations and perspectives:
The demonstrator is the results of the integration of several research project (walking, teleoperation, whole-body motion generation, planning, etc). Each of the modules is subject to its own limits and perspectives. In term of application, the main limitation is the balance of the humanoid robot, that should be able to robustly handle non-flat terrains and perturbations by the human collaborators before to be really used in industrial contexts.

12.2 Dance with a robot

Reference paper [2] (see also [19, 47])

The second demonstration answers to a command of Toulouse-City, asking for a mix technological and cultural know-how into a popularization event. For us, it was the opportunity to validate the use of the methods based on the inverse dynamics for quickly generating stable and efficient robot movements.

12.2.1 Design

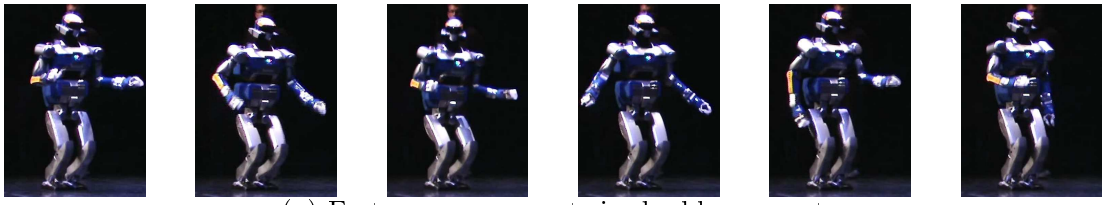
The demonstration was designed by the professional choreographer and dancer T. Benamara. The choreography of the robot was first executed by T. Benamara and recorded using the motion capture system. It was then cut in thirty-five sequences, each of them being systematically treated to generate the robot movements. Three types of sequence were considered: double-support (see Fig. 12.3-(a)), walking (see Fig. 12.3-(b)) and single-support (see Fig. 12.3-(c)&(d)) sequences. The generation of motion for each sequence type is very similar.

Consider the double-support motion class first. The human posture is retargeted to the closest robot configuration: each marker of the teacher is associated to a position on the robot body, which is adjusted by non-linear optimization. The position of the hands and the orientation of the gaze are extracted from the geometrical retargeting. The motion is then generated with an inverse-dynamics stack of tasks, with two contacts (one for each foot), and the following hierarchy of tasks: right-hand, left-hand, gaze (following the trajectories extracted from the demonstration) and finally the posture (adjusting the configuration at best to the posture of the human teacher). Depending of the choreography specifications, the hands and gaze tasks are relaxed if the movement is not meaningful for the dance. The COM is not constrained and the force constraints with the posture task are sufficient to ensure the balance.

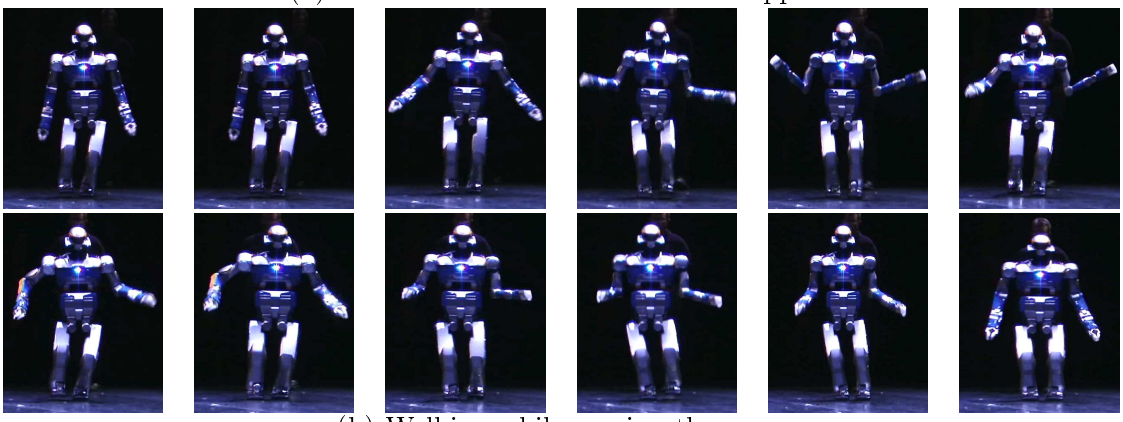
The walk is produced by a linearized inverse-pendulum pattern generator. The footprints and corresponding timings are first computed from the human trajectories, by detecting cluster of static points in the foot trajectories. From the footsteps, the COM and foot trajectories are computed by trajectory optimization [Morisawa 07, Herdt 10]. The motion of the whole body is generated as previously by an inverse-dynamics stack of tasks, whose contact points are one foot or the other (depending on the generated foot trajectories) and whose task hierarchy is: COM, flying foot (if any), right hand, left hand, gaze, posture.

The single-support sequences are generated similarly, with the COM trajectory being manually designed (a simple pattern generator could be used) and tracking the flying foot trajectory from the human performance.

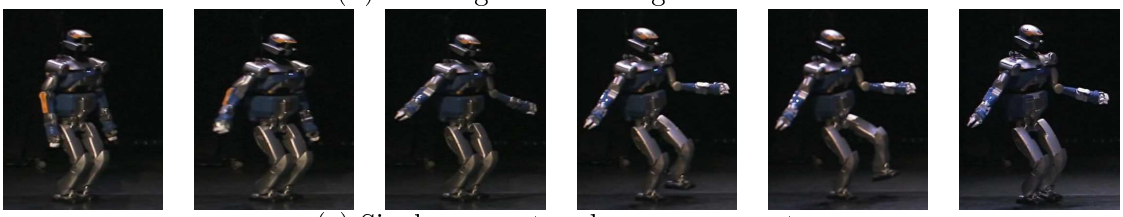
The motion is generated off line: the dynamic retargeting is nearly real time, but the data parsing and geometrical retargeting processes are not optimized and very slow. The whole process takes five times the real time. The motion is then replayed by the robot. The motion execution is nearly open loop. The only loop closure is done by the flexibility stabilizer [Kaneko 94, Kajita 10b]: each foot of the HRP-2 is equipped with a compliant sole that is meant to absorb the impacts during the walk. The control stabilizes the motion of these flexibilities using the leg motors, the foot force sensors and the reference trajectories of the leg joints, ZMP and chest attitude. The stabilizer is one of the basic controllers of HRP-2 and is used in nearly all our demonstrations.



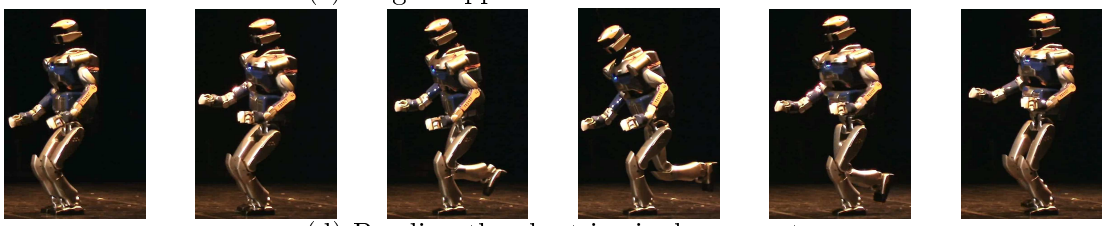
(a) Fast arm movements in double support



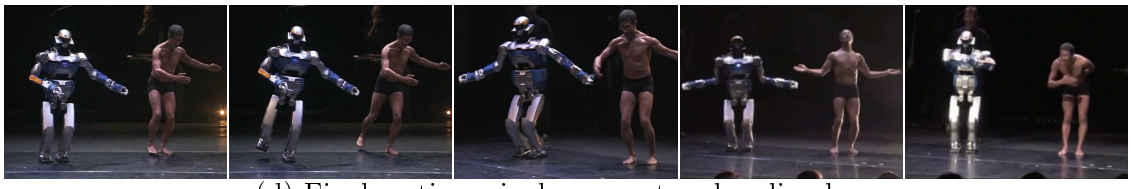
(b) Walking while moving the arms



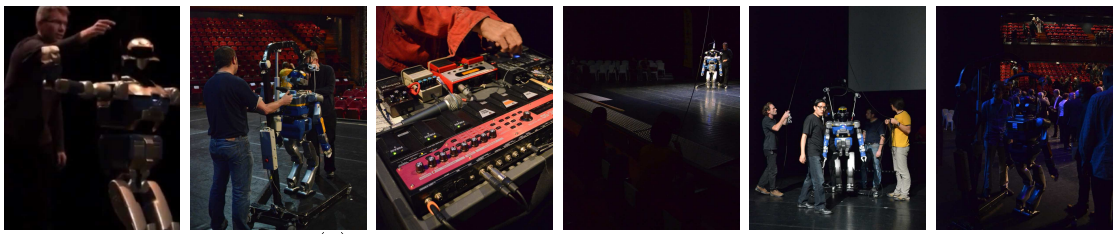
(c) Single support and arm movements



(d) Bending the chest in single support



(d) Final motion: single support and ending bow



(e) Snapshot of the event organization

Figure 12.3 – *Snapshot of the performance* "Dance with HRP2" *performed with T. Benamara using the HRP-2 robot, at Toulouse City music mall* "La Halle aux Grains"*, on the sixth of October, 2012.*

12.2.2 Results

The dace is composed of more than fifteen minutes of motion with nearly ten minutes generated from the motion capture. The rate of automatic translations from the human motion to a trajectory that the robot is able to replay is very high (100% for double-support sequences, 80% for single support) except for the walk. The generated upper-body motion implies important torques on the robot feet. The stabilizer is then not able to perfectly compensate for these dynamic effects, because of differences between the real robot and the model used by the control. Consequently, the robot behavior during the walk is poor and there is a higher risk of fall, which is not acceptable during a demonstration. For large motions of the arms during the walk, we finally had to simplify the choreography by removing the upper-body movements during some walk sequences (in four sequences over six).

These effects open some very interesting scientific problems in the control and estimation of the robot dynamics. Yet we are not able to apply the inverse dynamics proposed in Chapter 6 as a controller of the robot. The development of such a robust controller would make it possible to execute for example large arm movements during the walk. It is one of the perspectives developed in the next chapter.

As explained in Chapter 6, it is not possible to rely only of the instantaneously linearization to control the balance stability. Furthermore, it is difficult to track the movements of the human-teacher's COM. We rather had to rely on the external computation of the COM trajectory, which is disappointing. Using an approximation of the captured human COM (for example, the position of the hip bone) was not sufficient: apparently, the dynamics are too different to easily absorb such inaccuracy. None of that would be problematic if using a whole-body MPC optimizing the distance to the human posture on a preview window (for real-time imitation, obtaining a preview of the human posture would raise another kind of problem [Benallegue 10]).

12.2.3 Conclusions

The demonstrator proves the efficiency of the task-based robot programing approach. The complete dance show was composed of 15 minutes of whole-body movements and was developed by a just-graduated engineer (O. Ramos) in less than two months. The motion capture is of a great help in this task, as it is known for a long time in computer animation [Ginsberg 83, Sturman 94]. In particular, the only interaction of the choreograph with the motion generation system was through the motion capture, on the contrary to other robot dance where the choreograph has to work directly with the software defining the motion [Nakaoka 10]. With a little bit of additional engineering, it would be possible to provide to the choreographer an automatic and quick repetition of its movement by the robot, in order to enable a more fluent interaction during the design of the motion.

The strength of the approach comes in part from the use of the robot dynamics in the generation process. Contrary to inverse kinematics, where many parameters have to be tuned to enable a real execution by the robot (in particular to scale the respective velocities in the several task spaces and the accelerations at the beginning of each task), the inverse dynamics naturally produces movements fitting the robot dynamics and provides a better rate of success. According to us, the inverse dynamics is now nature enough to completely replace the inverse kinematics for motion generation on humanoid robots.

This success should not hide the fact that the robot follows a reference acceleration coming from the inverse-dynamics solver, while the reference torques would provide a more robust behavior during physical interactions (typically, for dealing with uncertain contacts like a non-flat floor). The inverse dynamics is seen here as a motion generation method rather than a controller. Moving to a real closed-loop sensor-based inverse-dynamics control onboard the real robot is one of the perspectives of this work and of a large community around the humanoid robotics today.

Even when only considering motion generation, a limitation of the approach is the incapacity to take into account the future of the robot. This is critical for considering the robot balance and for walking. The robot balance cannot be simply defined as a local criterion on the COM, like it is done with the quasi-static hypothesis in inverse kinematics. It is rather defined in terms of future evolution of the robot, for example by the viability [Wieber 02, Wieber 08] or capturability [Pratt 12, Koolen 12]. Today (and in particular for this demonstrator), we have to rely on a simplified model of the robot dynamics, the linearized inverse pendulum, to handle the future through trajectory optimization. It is not possible in these conditions to have a complete integration of walking and whole-body animation. As explained by J-P. Laumond in his class [Laumond 12], there is a gap between the underactuation inherent to the mobile robot and its hyper redundancy [Kanoun 11b]. The future could be taken into account by optimizing the whole future trajectory, as studied in Chapter 10, but the cost remains too expensive today for control, even if some new techniques could lead to this goal in a not-so-far future [Tassa 12, Pham 12].

On a technological point of view, the dance demonstrator is interesting for the short time necessary for its development, even if it does not reached the best performances, *e.g.* [Nakaoka 07]. However, for people at large, there is little differences and we encountered an unexpected interest during the live demonstration. For the first presentation, one unique live demonstration was planned, followed by a scientific debate. We finally had to cancel the debate and replace it by two additional demonstrations in front of a full room. The second year, the presentation (joined to a broader scientific debate with the scientific community of Toulouse) attracted more than 1000 persons. This interest, joined to the relatively quick development time (with the now-developed tools, we estimate the replication of such an event to three weeks of work for three persons including the choreographer) makes of live demonstrations with such robots a realistic business. Some professional demonstration robots already exists, such as the *"Dances with Robots"* in France in the Futuroscope [Solveig 12] or the *"Theater of Robots"* around the specific robot platform Robothespian [EAL 12].

12.3 What demonstrations teach us

This chapter was given as an exemplification of the work presented throughout this document. *"Robot@CWE"* is the occasion to challenge the versatility of our concepts into a broader demonstration involving a large architecture, several human partners and several computing units around the world. *"Dance with HRP-2"* validates the interest of the approach as a basic tools that a robotics engineer can use to animate the robot. These two demonstrations are also the occasion to emphasize the limits of the current work and are a transition to the perspectives developed in the next chapter.

Dance with HRP-2:
a case-study of fast development of dynamics whole-body movements

O. Ramos, N. Mansard, O. Stasse, S. Hak, L. Saab, C. Benazeth

Submitted to IEEE Robotics and Automation Magazine [2]

Context:
The paper presents the methodology used to develop the motions of HRP-2 for the dance show designed by the choreograph T. Benamara. The presentation occurred three times on October 2011 and once in October 2012 in front of 1200 spectators.

Motivations:
The *Dance with HRP-2* project is the opportunity to show the interest of the inverse-dynamics approach for the generation of whole-body movements for humanoid robots.

Approach:
The movements were first demonstrated by T. Benamara and recorded using a vision-based motion-capture system. The human posture is extracted by solving for each pose a non-linear least-square problem. The resulting posture is then dynamically retargeted using the inverse-dynamics solver. When necessary, tasks are added to obtain a more accurate replication of the movement (in particular, for fast motion of the COM). For walking, an inverse-pendulum based pattern generator is used to design the COM trajectory.

Results and contributions:
Motion capture together with inverse dynamics for reshaping allows a pretty fast development time. Nearly twenty minutes of motion were designed in less than two months by a single programmer.

Limitations and perspectives:
The process from the motion recording to the replay by the robot is not purely automatic: for complex movements, the robot programmer has to select some relevant aspects that are more important to imitate (e.g. hand position or COM). The motion is computed off line and then replayed in open loop by the robot. Real-time recognition and replication is a very demanding open challenge.

Sensing: The first aspect concerns sensor feedback. We have seen that humanoid robots are equipped with standard sensors, cameras, force sensors and IMU and that the task function approach enables to easily linked sensing and control, directly or by reconstruction. The sensory feedback is finally a simple adaptation of some techniques originally developed for classical manipulators (force control, visual servoing) or mobile robots (IMU-based localization tracking). However, we can wonder if the humanoid robot, by its mobility, underactuation and redundancy, does not imply the development of specific sensory capabilities. In the first demonstration, we can see that there are many different modalities during a robot action: manipulation, navigation in open terrains, walk with precise footstep positioning, manipulation while walking, etc. Each modality comes with some specific sensing model: navigation does not require the same accuracy of perception than manipulation. The originality of humanoid robots is the continuity between the different modalities. On a perfectly horizontal terrain, the robot can navigate like a unicycle. On a rough terrain [Hauser 08], the locomotion is rather resolved by the techniques developed for dexterous manipulation [Han 98, Bouyarmane 10]. The transition between flat ground and rough terrain is continuous and requires a continuous overlapping of sensory-feedback strategies.

These questions about sensing also cover the development of tactile capabilities (skin) [Dahiya 10] and variable impedance capabilities (*muscle-like* actuation) [Flacco 12]. In the same direction, there are some strong questionings about the sensor-based motion generation, control but also body design, for the control of the balance and the walk. These questions arise from the comparison with the human synergies, where some evidence shows that the control paradigms are in opposition to robot strategies.

Balancing: This last question opens to the sense of balance, for which most of the difficulty on the control part. Paradoxically in humanoid robotics, the balance is not a notion with a very strict definition. A given motion is often said stable when it can keep a stable contact [Pang 00, Hirukawa 06]. Typically, the walking motions obtained by MPC [Kajita 03] are said stable in the sense that the ZMP condition, upon which they are build, ensures a stable contact with some robustness [Sardain 04]. This notion is linked to balance if the robot implicitly is in a stable locomotion cycle [Grizzle 01]. More generically, balance is defined by the future states of the robot. Viability, which is the ability of not falling into a undesirable state [Wieber 02], is certainly the most generic definition. A reduction is capturability, the ability to nullify the robot velocity in a finite number of steps. [Pratt 12].

As said upper, it is not possible to take such complex criteria depending on the future states of the robot in the proposed control loop. We can hope that a direct writing of such conditions into a non-linear numerical solver would bridge the gap with the instantaneously-linearized formulation that we have considered here. In that case, the solution to the problem of balance is purely technological: we "just" have to wait for the sufficient computing capabilities and use them to solve a more complex but similar problem. A very impressive first step in this direction is given in [Tassa 12].

In the meantime, balance does not spontaneously appear in the motion generated by our approach. It has to be enforced by additional models, for example by artificially limiting the COM movements [Sugihara 02] or by pre-computing the COM trajectory using a walking pattern. This division between whole-body movements and balance criteria leads to artificial problems, such as the difficulty to choose if a walking step is necessary [Yoshida 06] or where the next footprint should be positioned [Kanoun 11b] to accomplish a criteria written at the

whole body level.

Planning: Finally, very little integration has been done at the planning level. In the first demonstration, all the decision capabilities have been forwarded at the symbolic level by the scheduler or directly to a human operator by using teleoperation. A planner was used but only for simple navigation. In the second demonstration, there was absolutely no decision, the demonstrator being rather at the motion-generation level. It is possible to autonomously plan movements with a humanoid robots [Kuffner 03, Yoshida 05, Berenson 11, Baudouin 11]. The problem is complex because of the dimension of the robot configuration space and of the manifold structure coming from the robot constraints [Dalibard 10]. This complexity is known and is studied.

Many other problems arise when trying to execute the planned trajectories on the real robot. Indeed, it is a difficult problem to formulate the planned trajectory as a control objective. The approach by joint trajectory is coming from the manipulation robotics, where the trajectory is the configuration space directly leads to a controller [LozanoPerez 83]. The same paradigm may not be the most useful in humanoid robotics. This issue is linked to the questions listed in the sensing paragraph on the previous pages: it is difficult to automatically find the sensor that can be used to servo the trajectory tracking. More generically, the problem comes that the planner does not give any explanation about the path it has selected. A semantic has to be recovered from the path before sending it to a controller.

These open questions are a mixed of technological and scientific problems. Some directions to address them are given in the next chapter.

Chapter 13

Conclusion and perspectives

Beyond the dream of a robot companion working as the perfect servant acting among humans, our laboratory humanoid robots are giving an idea of the potential application in robotics. For a direct application of the humanoid robot outside of the laboratories, safe locomotion and balance remain the major open issue, as illustrated by being one of the primary targets of the DARPA robotics challenge [Guizzo 12]. But humanoid robots also open the way to many developments for smart robotics cell and human-friendly mobile manipulators. In this view, the humanoid robot is only one complexity-step further to other robotics setup. All these setups, from the simple work cell in factory to the science-fiction among-human humanoid robot, are needed some advances in soft motion prototyping, for both supervised and autonomous motion generation. In this chapter, my contributions to achieve this goal are first quickly summarized, before introducing my future research orientations.

Contributions

The document has followed our bottom-up approach. Contributions have been detailed at the end of each chapter. A quick list is synthetically recalled here.

Part I: Control level

The first part is aiming at building a task-based low level controller that simultaneously covers all the robot modalities and offer some easy *high-level* control inputs. The approach relies on the task-function approach. It provides an abstraction hiding the joint-level robot control and in exchange offers a proto-symbolic control of the robot by selecting or modifying the active tasks.

Hierarchical task inversion: The main concern when combining several tasks on a redundant structure is the possible arising singularities. Singularities are intrinsically due to the

non linearities coming from the robot structure. However, if each combination of two tasks may rise a singularity due to possible conflict, the number of singular regions exponentially increases with the number of tasks. Setting a hierarchy between the active tasks is a way to handle the singularities due to conflict, or at least a way to limit the impact of each singularity to a limited number of DOF. Based on the well-known works on the redundancy framework [Siciliano 91, Baerlocher 04], our work has focused on the insertion of inequality-written tasks into the stack of tasks. We have proposed two solutions. The first one [8] is based on a homotopy between free and saturated controllers. It offers a very smooth control. However, the cost increases with the number of inequality constraints and prevents practical systems with a large number of DOF. The second one [4] is based on active-search algorithms, which are a way to handle this complexity. We have proposed a hierarchic version of the active-search algorithm and implemented it into an efficient HQP solver [28, 4]. The solver is able to solve in real-time (less than 1ms) an inverse kinematics for a 40-DOF robot. In my opinion, this solver and its application to inverse dynamics is the main practical contribution of this document. I believe it is going to replace any iterative pseudo-inverse based inverse-kinematics implementations (that we have shown theoretically equivalent and far less efficient) in a near future.

Continuity and damping: The hierarchy is a way to handle the conflicts between tasks, by preventing the propagation of the singularity effects to the tasks having priority. The hierarchy also simplifies the diagnostic by isolating the singular task. We have shown how it is possible to put into evidence the source of the numerical problems from the HCOD at the foundation of the solver. The next step is to automatically treat the singularity, by adding a kind of "*fuse*" that would continuously cut out the singular parts of the task. Such a fuse is known as *damping* by roboticists. It avoids a waste of motion resources by a task that, in any case, will not be properly executed because singular.

If we are not able to propose a solution yet, our contribution here is to make explicit the need of such a damping term and to connect it to the problem of continuous task sequencing. Indeed, removing a task from the stack is nothing more than increasing its damping term until effective cancellation. Practical solutions to enforce the control continuity during the task transition have also been proposed.

Fast operational-space inverse dynamics: Inverse kinematics can only handle motions in the free space. Contact or balance can be artificially introduced but this formulation presents some limitations. The natural extension is to add the dynamics model of the robot. We have proposed an operational-space inverse dynamics able to handle priorities among a large number of tasks. The objective is to reach real-time (1ms) computation on board robot computers. With the proposition of a condensed dynamic solver, we nearly reached this objective and have started the practical implementation on the robot.

And the robot: A big effort has been deployed to effectively applied the methods on the robot. The physical machine brings uncertainties in the control loop. These uncertainties are resolved by closing it on the sensor feedbacks. We have in particular worked with the cameras, the force sensors and the IMU to produce real and efficient movements on the humanoid robot HRP-2.

Part II: Emergence of a motion semiotics

The second part of the document took advantage of the low level layer built in the first part to propose a methodology to assemble the basic motion components in complex movements. Rather than a complete working method, this second part proposes a promising route for future researches that we refer as "semiotics of motion".

Automation of task sequences: The ultimate goal is to provide an automatic solution to assemble the tasks in a complex sequence. In this direction, we have shown that a task sequence can be used as an implicit trajectory representation. Such a representation enables at the same time a planner to perform geometrical reasoning on the trajectory and a controller to efficiently execute the trajectory while relying on the sensors. We have proposed a solution to compute the numerical components of a task and automatically convert a deliberative symbol into a control-ready proto-symbol.

Motion description: Since a task can be used to represent a motion to be executed by the robot, it can also be used to describe an observed motion. We have used decomposition by tasks to recognize the movement realized by an external structure. The method is able to distinguish between very similar-looking movements.

And the robot: We also made a lot of effort to provide an efficient implementation of our approach that can be used to animate a real robot. In particular, the software *StackOfTasks*, counting about 250 000 lines of code, is provided in open source and has been used in several demonstrations and experiments in four (and soon five) laboratories. We have also worked directly in two real-scale demonstrators. The first one, *Robot@CWE*, proves the concept of semi-autonomous humanoid robots working in CWE. The second, *Dance with HRP-2*, demonstrates the expressivity of the task-based approach to rapidly developed complex robot motion.

Perspectives

Several perspectives have been expressed along the document, in particular at the end of each chapter. Some of them are gathered here and developed into a wider perspective. A direct continuation of the work presented in the first part is necessary to reach a complete embodiment of the multiple robot modalities. The objective is to build the larger possible vocabulary of action, equipped with powerful and generic combination rules that make the composition of complex action possible. In particular, perspectives are open to go one step lower and use specific mechatronics to handle the limitation of the control.

This action vocabulary can be at the origin of two explorations. First, as explained by the preliminary experiments we did, the task, viewed as proto-symbols of movement, can be used to bridge the gap between planning and control. This does not solve the intrinsic complexity of planning but gives a way to link geometrical reasoning to control supervision.

The developed low-level functionalities are progressively bringing the humanoid robot closer to the humanoid body capabilities. The tools that we are developing to animate the

robot can also be used to describe the movement principles of human beings. Direct applications to the field of artificial movements are the realistic animation of avatars, in particular for ergonomics study on virtual prototypes, and learning-by-observation to ease the robot programming.

Whole-body sensor-based dynamics-consistent model-predictive control of the humanoid robot

This title looks like a Christmas wish list. It however corresponds to the real todo-list to complete before completely covering the capabilities of the robot. In particular, the dynamics motion produced in Chapter 6 were generated off-line, and there is a lot of work to do before obtaining the same result on the robot while taking into account the sensor feedback. The creation of such sensory-motor loops requires to take into account or to develop specific hardware for both the actuation and the perception systems. A preview window on the robot future should be considered while computing the control law. MPC is particularly important when the dynamics play a central role, for balance or when considering elastic materials. These three aspects are closely related and are given below in an arbitrary order.

Sensor-based dynamics: The target achieved in Chapter 6 was to compute the operational-space inverse dynamics in real time. With classical CPU (*e.g.* those embedded at reasonable cost on the HRP-2 robot), we achieve the whole-body computation in 5ms. For a method of similar complexity [de Lasa 10], 1kHz was achieved by increasing the cost (both financial and energetic) of the CPU. These results open the possibility to use the inverse-dynamics scheme as the controller of the robot. Several problems need to be considered. First, an inverse-dynamics scheme is relevant if the loop is closed on the force data. On the other humanoid robot models of the same generation than HRP-2, only position control is possible. One solution is to identify the actuation model and to *by-pass* the position-based controller [Khatib 08]. On the other hand, it is possible to rather send the desired acceleration (integrated twice into a desired position) and rely on force sensing to correct the impedance loop [Hogan 85]. Both approaches are known to be limited, but can enable to obtain preliminary results, in particular concerning the walk on uncertain terrains.

Following the technology developed in the HRP, Asimo or Hubo robots, France has started some original design with cable backdrivable actuators [Olaru 09, Guizzo 10] that might enable position controller to execute dynamic movements with a torque feedback of every joints. Romeo is expected to settle in our laboratory in Summer 2013.

The alternative is to completely change the technology and to use actuators with intrinsic torque-based control capabilities, by adding effort gauge following the DLR developments [Ott 10] or with compliant actuation [Vanderborght 10]. It can be remarked that HRP-2 already has such capabilities, by embedding in the ankle at the same time a flexible element and a 6D force-torque sensor. For walking at least, we have the proper experimental setup.

Consider now that it is possible to execute with the robot a torque reference. The question is then to be able to reconstruct some models of physical interfaces. Preliminary worked

have started with manipulators arm [Park 06b, Petrovskaya 07]. The approach has to be generalized and augmented by the feedback of other sensors: for example adding vision to force to reconstruct a surface, or the IMU to the integral of force data to evaluate the ground friction or elastic parameters.

Toward dedicated mechatronics: The application of dynamics-based sensorimotor loops are closely linked to the design and integration of new dedicated hardware. The insertion of joint-torque sensing capabilities implies the addition of flexibilities in the kinematic tree. It makes more complex the control models but it also enables the robot to store some energy in the flexible parts and increase its power.

Force sensors measure the physical effects acting of the robot structure. For reconstructing the contact interface, tactile capabilities (such as offered by a skin) provides very complementary information. For example, for point contact, a skin enables to localize the contact points, while a single force sensor needs to combine several measures on a time interval before reconstructing it.

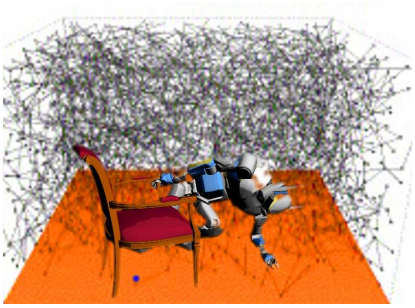
Finally, if the humanoid tree shape is (by definition) nearly fixed, the design of specific kinematic or actuation structures would be a need for efficient robotics solution. The design parameters (body length or mass) are already often decided by numerical optimization. We could hope to integrate the mechanical design to the motion generation process to ease and improve the decision of an integrated solution.

Preview window: As we repeated several times in the document, the instantaneous linearization, done to obtained a simpler problem and track it in real time, is limited. The robot balance cannot be considered as an instantaneous criterion. Similarly, the movements of flexible elements are not subject to instantaneous linearization. Two concurrent approaches can be considered.

The first one is to follow the hierarchical approach used for walking [Kajita 03]: a specific reduced model is extracted, that gathers the relevant dynamical effect. This simplified model is handled by MPC and the solution is injected in the instantaneous linearized model that then handles the whole-body aspects. This approach can typically be applied at the level of each joints to handle the flexibilities [AlbuSchäffer 07b, Sardellitti 12]. The two drawbacks are the lack of genericity (a dedicated model has to be design for each relevant dynamics) and its sub-optimality (each dynamics being handled separately, which prevents any synergy between them).

On the other hand, the entire dynamics can be solved in a big optimization problem. The system is huge, but the theoretical continuity property of the optimum with respect to the problem changes can be used to warm start each resolution and to significantly reduce the cost. Still, until recently, this direction was considered to be very appealing but also very exploratory. The results demonstrated in [Tassa 12] show that it is now technologically possible or nearly possible to apply this solution. The preview window widely increases the expressivity of the control. It would for example directly handle the flexible elements. However, prediction of the control means that we also have to predict the evolution of the sensors, *i.e.* to increase the accuracy of the environment models, which can be difficult to perform in practice.

Unifying planning and control



The idea behind the task-based control approach is to absorb at the lowest possible level the maximum of uncertainties. Consequently, it is possible to reduce the granularity of the model used at higher levels. However, the objective is not to hide to a hypothetical symbolic level the physical world that is intrinsically numeric. On the opposite, the task-based level provides the necessary inputs to act in this numerical world. We have shown in Chapter 10 that the task sequence can be used to describe a trajectory simultaneously for the planner and the controller. The developments of this idea are to use it to efficiently drive some geometrical reasoning.

From folded configuration space to symbols: We have described the specific *folded* structure of the configuration space of walker robots [Siméon 04]. Each leaf corresponds to a set of contacts along with the robot is moving. Locomotion (at large) can be achieved by moving from one leaf to the other. The classical complexity of path planning, arising from the robot dimension, is increased by this nearly symbolic structure. The contact constraint is in fact very close to the notion of task. When a task is regulated, it imposes a fixed point in a given submanifold; said differently, it creates an implicit leaf in the configuration space. Moving toward the regulation of a task is nothing more than reaching a specific leaf along with a new action can be activated. Sequences of tasks and contacts are used to solve the manipulation problem in [Siméon 04].

The same approach can be generalized to various situations for simultaneous humanoid locomotion and manipulation and in other simpler contexts of robotics. However, when considering actions triggered by other actions or by a given object position, the problem dimension increases and so increases the complexity. For example, if the robot can pick up an object after opening a drawer, the state of the drawer (open or close) as to be embedded in the configuration space and increases the dimension. Hierarchical reasoning coming from deliberative process can then be used to hide this complexity.

Task-based trajectories: To reduce the complexity, the planner works with simplified models. For example, kino-dynamics planning is barely used for humanoid robotics: the model generally neglects the dynamics, or is reduced to the dynamics to a free floating wheel [Dellin 12]. The output trajectory in the configuration is thus implicit. Optimization can be used to validate or locally correct the whole-body trajectory [El Khoury 13, Mordatch 12].

In that case, the task-based trajectory is interesting. Such a representation does not need the complete configuration trajectory to be computed. It also gives an augmented robustness, since the trajectory is defined as an implicit tube and not as a zero-measure thread. In particular, the vector field controlling the robot to track the trajectory is explicitly (or nearly explicitly when tasks have to be combined) given by the task definition. However, the convergence domain of the tube is not infinite and its bound are difficult (not to say impossible) to compute. Moreover, the sequence of tasks corresponds to a physical meaning: there is no sense to try to open the fridge door if the handle has not yet be reached. The control is then finally rewritten using the classical symbolic execution paradigms and the

coupling with the sensor-based control has to be considered.

Synergies: Finally, we can deduce from simple observation that complex geometrical reasoning is most of the time avoided by human or other intelligent animals. When necessary, the search is well guided by the brain experience. In [Dalibard 10], specific structures are added to describe the affordance of the objects located in the robot environment, to guide the search and to simplify the planning. Such *documentations* could be increased to gather the main part of the decision process when the planner encounters a classical situation.

A large part of the complexity of sampled-based planning comes from the connection tests to validate the transition between two nodes. There is a trade-off to find when building the planner: a more complex local method has a better chance to connect two nodes but is more costly to evaluate. Learning techniques can be developed to approximate the feasibility function and only test the connections that really have a chance to be validated [Perrin 10]. In the simplest cases, the feasibility function can validate an action without even calling the planner.

Finally, empirical combination of actions can be stored and used to improve the performances or the robustness of a movement. For example, when moving the arm to reach an object in a cluttered environments, risks of collision are lowered by keeping the hand in the FOV [Kanoun 09a]. Such synergies are even more important when using a redundant actuation chain to perform a movement with a high dynamics, where each motor has to be used at best to cope with the robot power limits. Learning from experience can be used to build a description of the synergies. Their insertion in a task-based planner or in a task-sequence controller is straightforward. Strong connections can be made with human motion description, as discussed in the next section.

Toward human motion representation

The previous two sections have focused on motion generation, planning and control in robotics. In particular, we have concluded on the possibility to learn from human strategies. We now consider the dual perspectives: "how can the tools developed for robotics be used to explain the natural motion?". At the first level, we can only consider the human body and try to replicate the way redundancy (both in motion and actuation) is handled. Our approach can also renew a dialog with movement neuroscience.

Biomechanics: The human body can be modeled by a rigid kinematic tree, subject to joint constraints and limits, and actuated by redundant linear actuators, the muscles, wired to two or more links by tendons. The muscles have their own constraints, in particular they are limited to positive effort, to which it is possible to add constraints on the muscle or group of muscles activation. Accurate models exist that directly enter in our motion-generation framework. A first attempt would be to use these methodologies to generate in simulation a motion of the musculoskeletal model. Such a result has direct applications for example in PLM for virtual ergonomics study.

The musculoskeletal model is redundant, both in motion and in actuation: for a same end-effector target, there are several joint acceleration and muscle forces performing the reference. The next step is to recognize the strategies used by the human body to select among this redundancy. We can use numerical learning to identify these strategies, in order to replicate them, for ergonomics purposes or realistic human motion rendering in computer animation.

Behind the numerical replication, understanding these synergies and adapting them to redundant robot would close the loop, from robot to biomechanics and back to the robot.

Neuroscience of motion: From the robotics study, we gather mathematical and engineering tools to describe the motion. The biomechanics gives us a model of the body implementing the embodiment of the movement. The natural continuation is to search for the control models of the human movement. The representation of action is a key research issue today in neuroscience. Our tools for studying and generating artificial movements can give an input to this research and lead to beneficial dialogs in both directions.

Such a nice robot!

I would like to insist a last time about the importance in robotics of the application on a physical robot acting in the real world. This aspect becomes even more important now that robotics is slowly moving to the real world. One of the major perspectives is finally nothing more than the direct application of our methods to industrial problems. The first objective of the task-based approach is to simplify the programing of the robot motion and by this way, to ease the introduction of complex robots in versatile processes. A typical example is the concept of a *Factory in a Day* [Wisse 13], that is to say a generic industrial robotics process that can be deployed within a day to start a specific production. Our robots are redundant, in the sense that they can perform multiple variety of tasks without hardware modification, by simply changing the software. What is really missing now is to have the proper software toolboxes to demonstrate the interest and the readiness of such technologies outside of the laboratory context.

Appendix

Appendix A

Generalized inverse

A.1 Moore-Penrose pseudoinverse

The Moore-Penrose pseudoinverse can be seen as a generalization of the matrix inverse when the matrix is not invertible. It is uniquely defined by the four conditions:

$$AA^+A = A \tag{A.1}$$

$$A^+AA^+ = A^+ \tag{A.2}$$

$$AA^+ \text{ is symmetrical} \tag{A.3}$$

$$A^+A \text{ is symmetrical} \tag{A.4}$$

It is referred as the pseudoinverse in all the document. A^+ is also called the least-square inverse because it solves the min-min problem

$$A^+b = \min_{x \in \mathcal{S}^*} ||x|| \tag{A.5}$$

with

$$\mathcal{S}^* = \min_x ||Ax - b|| \tag{A.6}$$

In other world, A^+b corresponds to the least-square x that minimizes the distance from Ax to b.

A.2 Non constructive property

The pseudoinverse always respects the following property:

$$A^+ = A^T(AA^T)^+ = (A^TA)^+A^T \tag{A.7}$$

When A is full-row rank (*i.e.* surjective, the rows of A are a free vector family), then AA^T is invertible and the first equality can be rewritten as a constructive property:

$$A^+ = A^T(AA^T)^{-1} \tag{A.8}$$

When A is full-column rank (*i.e.* injective, the columns of A are a free vector family), then A^TA is invertible and the second equality can be rewritten as a constructive property:

$$A^+ = (A^TA)^{-1}A^T \tag{A.9}$$

In general, these two equalities are false and only (A.7) is ensured. Moreover, if the coefficients of A^+ have to be computed, non of the two forms above should be used as they involved two matrix products and one poorly-conditioned inversion. Indeed, A^TA and AA^T norms are lower than the square of A norm. Any numerical algorithms used to invert them would then behave poorly.

A.3 Other generalized inverses

In the general case, the four conditions are necessary and sufficient to uniquely define the inverse. Inverses can be defined that only respects some of the four Moore-Penrose conditions.

To ensure a good behavior of the pseudoinverse-based control laws, (A.1) and (A.2) are often sufficient[1]. A class of inverses that do not respects (A.3) and (A.4) are the weighted inverse [Doty 93]. The size of A is denoted by $m \times n$. Consider two positive definite *weight* matrices L and R of dimension $n \times n$ and $m \times m$, respectively. These matrices define new norms $||x||_L = \sqrt{x^TLx}$ and $||y||_R = \sqrt{y^TRy}$. Then the R-right L-left weighted inverse $A^{L\#R}$ of A is defined as the unique matrix that ensures a minimal R-norm in $\mathbb{R}^m$ and a minimal L-norm in $\mathbb{R}^m$ [BenIsrael 03] *i.e.*

$$A^{L\#R}b = \min_{x \in \mathcal{S}_L^*} ||x||_R \tag{A.10}$$

with

$$\mathcal{S}_L^* = \min_x ||Ax - b||_L \tag{A.11}$$

Denoting by $\sqrt{L}$ (resp. R) any matrix such that $\sqrt{L}\sqrt{L}^T = L$ (for example, the Cholesky decomposition of L [Golub 96]), the weighted inverse can be constructed by:

$$A^{L\#R} = \sqrt{R}(\sqrt{L}A\sqrt{R})^+\sqrt{L} \tag{A.12}$$

In particular, when A is full row rank, $A^{L\#R}$ is independent of L (since the least-square residue is always null). It is denoted by $A^{\#R}$. Using (A.8), the weighted inverse can be constructed from [Doty 93]:

$$A^{\#R} = RA^T(ARA^T)^{-1} \tag{A.13}$$

when A is full column rank, $A^{L\#R}$ is independent of R (since A has a trivial null space). It is denoted by $A^{L\#}$. Using (A.9), the weighted inverse can be constructed from [Doty 93]:

$$A^{L\#} = (A^TLA)^{-1}A^TL \tag{A.14}$$

Right weighted inverse are very important in inverse dynamics. R is then chosen to be the generalized inertia matrix of the system. Left weighted inverse are important for varying-feature-set tasks, where L is often used as an activation matrix.

[1] A matrix satisfying (A.1) and (A.2) is called a reflexive generalized inverse

A.4 Singularities

A singularity arises when local structure of the direct geometry map h collapses, *i.e.* when the rank of the corresponding Jacobian decreases. The pseudoinverse is always defined, thus the behavior inside the singular region is not problematic. The pseudoinverse has a similar behavior to the function:

$$\sigma \to \begin{cases} \frac{1}{\sigma} & \text{if } \sigma \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.15}$$

This is actually the pseudoinverse of the matrix $\begin{bmatrix}\sigma\end{bmatrix}$.

The point 0 is singular. Far from 0 and at exactly 0, the behavior is correct. The problem is in the neighborhood of the singularity: there the inverse map increases toward the infinity.

A classical solution is to *damp* the behavior of the inverse. This corresponds to apply a Tikhonov regularization of the QP (3.17):

$$\min_{\dot{q}} ||J\dot{q} - \dot{e}^*||^2 + \eta^2||\dot{q}||^2 \tag{A.16}$$

where η is the damping parameter. The damped inverse can be explicitly formulated by:

$$J^{\eta\dagger} = J^T(JJ^T + \eta^2 I)^{-1} \tag{A.17}$$

The damping of (A.15) is:

$$\sigma \to \frac{\sigma}{\eta^2 + \sigma^2} \tag{A.18}$$

This is equivalent to $\frac{1}{\sigma}$ when σ is large and equivalent to $\frac{\sigma}{\eta^2}$ when σ is close to 0. The damped inverse will thus behaves like the pseudoinverse far from the singularity borders and like the Jacobian transpose $\frac{1}{\eta^2}J^T$ close to the singularity border.

Bibliography

[Al Borno 13] M. Al Borno, M. de Lasa, A. Hertzmann. – Trajectory optimization for full-body movements with complex contacts. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1, 2013.

[AlbuSchaffer 07a] A. Albu-Schaffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimbock, G. Hirzinger. – The dlr lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: An International Journal*, 34(5):376–385, 2007.

[AlbuSchäffer 07b] A. Albu-Schäffer, C. Ott, G. Hirzinger. – A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *Int. Journal of Robotics Research*, 26(1):23–39, 2007.

[Allibert 10] G. Allibert, E. Courtial, F. Chaumette. – Predictive control for constrained image-based visual servoing. *IEEE Trans. on Robotics*, 26(5):933–939, 2010.

[Anjos 12] M. Anjos, J-B. Lasserre. – *Handbook on Semidefinite, Conic and Polynomial Optimization.* – Springer, 2012.

[Arisumi 08] H. Arisumi, S. Miossec, J-R. Chardonnet, K. Yokoi. – Dynamic lifting by whole body motion of humanoid robots. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'08)*, Nice, France, September 2008.

[Au 09] S. Au, J. Weber, H. Herr. – Powered ankle–foot prosthesis improves walking metabolic economy. *IEEE Trans. on Robotics*, 25(1):51–66, 2009.

[Baerlocher 04] P. Baerlocher, R. Boulic. – An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 6(20):402–417, August 2004.

[Baillie 04] J. Baillie. – Urbi: A universal language for robotic control. *International Journal of Humanoid Robotics*, pp. 7–29, 2004.

[Baker 09] K. Baker, D. Trietsch. – *Principles of Sequencing and Scheduling.* – Wiley Publishing, 2009.

[Barraquand 92] J. Barraquand, B. Langlois, J.-C. Latombe. – Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):224–241, March 1992.

[Basile 12] F. Basile, F. Caccavale, P. Chiacchio, J. Coppola, C. Curatella. – Task-oriented motion planning for multi-arm robotic systems. *Robotics and Computer-Integrated Manufacturing*, 28(5):569–582, 2012.

[Baudouin 11] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, E. Yoshida. – Real-time replanning using 3d environment for humanoid robot. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'11)*, Bled, Slovenia, October 2011.

[Benallegue 10] M. Benallegue, P-B. Wieber, A. Kheddar, B. Espiau. – A computational model for synchronous motion imitation by humans: The mirror controller applied on stepping motions. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'10)*, Nashville, USA, October 2010.

[BenIsrael 03] A. Ben-Israel, T. Greville. – *Generalized inverses: theory and applications.* – Springer, 2nd edition, 2003.

[Berenson 11] D. Berenson, S. Srinivasa, J. Kuffner. – Task space regions: A framework for pose-constrained manipulation planning. *Int. Journal of Robotics Research*, 30(12):1435 – 1460, October 2011.

[Besseron 08] G. Besseron, C. Grand, F. Ben Amar, P. Bidaud. – Decoupled control of the high mobility robot hylos based on a dynamic stability margin. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'08)*, Nice, France, September 2008.

[Bobrow 98] J. Bobrow, B. McDonell. – Modeling, identification, and control of a pneumatically actuated, force controllable robot. *IEEE Trans. on Robotics and Automation*, 14(5):732–742, 1998.

[Bock 83] H. Bock, K-J. Plitt. – A multiple shooting algorithm for direct solution of optimal control problems. – *Proceedings of the 9th IFAC World Congress*, Budapest, Hungary, 1983.

[BostonDynamics 13] BostonDynamics. – Dynamic robot manipulation. – youtube ID 2jvLalY6ubc, February 2013. (http://youtu.be/2jvLalY6ubc).

[Boulic 92] R. Boulic, D. Thalmann. – Combined direct and inverse kinematic control for articulated figure motion editing. *Computer Graphics Forum*, 11(4):189–202, August 1992.

[Bouyarmane 10] K. Bouyarmane, A. Kheddar. – Static multi-contact inverse problem for multiple humanoid robots and manipulated objects. – *IEEE-RAS Int.*

Conf. on Humanoid Robots (Humanoid'10), Nashville, USA, October 2010. – Best paper award finalist.

[Bouyarmane 11a] K. Bouyarmane. – *De l'Autonomie des Robots Humanoïdes : Planification de Contacts pour Mouvements de Locomotion et Tâches de Manipulation.* – Montpellier, France, PhD. Thesis, Univ. Montpellier, November 2011. A. Kheddar supervising.

[Bouyarmane 11b] K. Bouyarmane, A. Kheddar. – Multi-contact stances planning for multiple agents. – *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[Boyd 04] S. Boyd, L. Vandenberghe. – *Convex Optimization.* – Cambridge University Press, 2004.

[Boyd 07] S. Boyd, B. Wegbreit. – Fast computation of optimal contact forces. *IEEE Trans. on Robotics*, 23(6):1117–1132, December 2007.

[Bretl 08] T. Bretl, S. Lall. – Testing static equilibrium for legged robots. *IEEE Trans. on Robotics*, 24(4):794–807, August 2008.

[Bruyninckx 00] H. Bruyninckx, O. Khatib. – Gauss' principle and the dynamics of redundant and constrained manipulators. – *IEEE Int. Conf. on Robotics and Automation (ICRA'00)*, San Fransisco, USA, April 2000.

[Buehler 05] M. Buehler, R. Playter, M. Raibert. – Robots step outside. – *International Symposium on Adaptive Motion of Animals and Machines*, Ilmenau, Germany, September 2005.

[Bussy 12] A. Bussy, A. Kheddar, A. Crosnier, F. Keith. – Human-humanoid haptic joint object transportation case study. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'12)*, Lisbon, Portugal, 2012. – (video http://www.youtube.com/watch?v=kYUdYOIYeZ0).

[Caccavale 08] F. Caccavale, P. Chiacchio, A. Marino, L. Villani. – Six-dof impedance control of dual-arm cooperative manipulators. *IEEE/ASME Transactions on Mechatronics*, 13(5):576–586, 2008.

[Carloni 12] R. Carloni, B. Vanderborght, A. Albu-Schäffer, A. Bicchi (Edts). – *Variable Stiffness Actuators moving the Robots of Tomorrow.* – St Paul, USA, IEEE ICRA'12 workshop, April 2012.

[Chadwick 09] E. Chadwick, D. Blana, A. van den Bogert, R. Kirsch. – A real-time, 3-d musculoskeletal model for dynamic simulation of arm movements. *IEEE Transactions on Biomedical Engineering*, 56(4):941–948, 2009.

[Chang 95] T. Chang, R. Dubey. – A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators. *IEEE Trans. on Robotics and Automation*, 11(2):286–292, April 1995.

[Chaumette 01] F. Chaumette, E. Marchand. – A redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing. *IEEE Trans. on Robotics and Automation*, 17(5):719–730, October 2001.

[Chaumette 06] F. Chaumette, S. Hutchinson. – Visual servo control (part i). *IEEE Robotics and Automation Magazine*, 13(4):82–90, 2006.

[Cheah 05] C. Cheah, D. Wang. – Region reaching control of robots: Theory and experiments. – *IEEE Int. Conf. on Robotics and Automation (ICRA'05)*, pp. 986–992, Barcelona, Spain, May 2005.

[Cheah 07] C. Cheah, D. Wang, Y. Sun. – Region-reaching control of robots. *IEEE Trans. on Robotics*, 23(6):1260–1264, December 2007.

[Chestnutt 09] J. Chestnutt, Y. Takaoka, K. Suga, K. Nishiwaki, J. Kuffner, S. Kagami. – Biped navigation in rough environments using on-board sensing. – St Louis, USA, October 2009.

[Chevallereau 01] C. Chevallereau, Y. Aoustin. – Optimal reference trajectories for walking and running of a biped robot. *Robotica*, 19(5):557–569, 2001.

[Chiaverini 97] S. Chiaverini. – Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Trans. on Robotics and Automation*, 13(3):398–410, June 1997.

[Claeys 12] M. Claeys, D. Arzelier, D. henrion, J.B. Lasserre. – Measures and lmi for impulsive optimal control with applications to space rendez vous problems. – *American Control Conference (ACC'12)*, Montréal, Canada, 2012.

[Collette 07] C. Collette, A. Micaelli, C. Andriot, P. Lemerle. – Dynamic balance control of humanoids for multiple grasps and noncoplanar frictional contacts. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'07)*, Pittsburgh, USA, December 2007.

[Comport 06] A. Comport, E. Marchand, F. Chaumette. – Statistically robust 2d visual servoing. *IEEE Trans. on Robotics*, 22(2), April 2006.

[Dahiya 10] R. Dahiya, G. Metta, M. Valle, G. Sandini. – Tactile sensing – from humans to humanoids. *IEEE Trans. on Robotics*, 26(1):1–20, January 2010.

[Dalibard 09] S. Dalibard, A. Nakhaei, F. Lamiraux, J.P. Laumond. – Whole-body task planning for a humanoid robot: a way to integrate collision avoidance. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'09)*, Paris, France, December 2009.

[Dalibard 10] S. Dalibard, A. Nakhaei, F. Lamiraux, J.P. Laumond. – Manipulation of documented objects by a walking humanoid robot. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'10)*, Nashville, USA, October 2010.

[Dalibard 11] S. Dalibard, A.E. Khoury, F. Lamiraux, M. Taix, J.P. Laumond. – Small-space controllability of a walking humanoid robot. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'11)*, Bled, Slovenia, October 2011.

[Dantam 13] N. Dantam, M. Stilman. – The motion grammar: Analysis of a linguistic method for robot control. *IEEE Trans. on Robotics*, 2013. – [in press].

[Dautenhahn 05] K. Dautenhahn, S. Woods, C. Kaouri, M. Walters, K. Koay, I. Werry. – What is a robot companion – friend, assistant or butler? – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'05)*, Edmonton, Canada, August 2005.

[DaVinci 19] L. Da Vinci. – *Codex Atlantica*. – 1519. (from Leonardo's Lost Robots, M. Rosheim, Springer, 2006).

[Davison 07] A. Davison, I. Reid, N. Molton, O. Stasse. – Monoslam: Real-time single camera slam. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[de Lasa 10] M. de Lasa, I. Mordatch, A. Hertzmann. – Feature-based locomotion controllers. – *ACM SIGGRAPH'10*, July 2010.

[Decré 09] W. Decré, R. Smits, H. Bruyninckx, J. De Schutter. – Extending iTaSC to support inequality constraints and non-instantaneous task specification. – *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan, May 2009.

[Dellin 12] C. Dellin, S. Srinivasa. – A framework for extreme locomotion planning. – *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, Saint Paul, USA, May 2012.

[DeLuca 97] A. De Luca, G. Oriolo, C. Samson. – Feedback control of a nonholonomic car-like robot. *Planning Robot Motion*, J-P. Laumond Edts., chap. 4. – Springer-Verlag, 1997.

[DeLuca 06] A. De Luca, A. Albu-Schäffer, S. Haddadin, G. Hirzinger. – Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, Beijin, China, October 2006.

[Demircan 09] E. Demircan, J. Wheeler, F. Anderson, T. Besier, S. Delp. – Emg-informed computed muscle control for dynamic simulations of movement. – *XXII Congress of the International Society of Biomechanics*, Cape Town, South Africa, July 2009.

[Deo 92] A. Deo, I. Walker. – Robot subtask performance with singularity robustness using optimal damped least squares. – *IEEE Int. Conf. on Robotics and Automation (ICRA'92)*, pp. 434–441, Nice, France, May 1992.

[Diehl 09] M. Diehl, H. Ferreau, N. Haverbeke. – Efficient numerical methods for nonlinear mpc and moving horizon estimation. *Nonlinear Model Predictive Control*, pp. 391–417, 2009.

[Diftler 11] M. Diftler, J. Mehling, M. Abdallah, N. Radford, L. Bridgwater, A. Sanders, R. Askew, D. Linn, J. Yamokoski, F. Permenter et al. – Robonaut 2-the first humanoid robot in space. – *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[Doty 93] K. Doty, C. Melchiorri, C. Bonivento. – A theory of generalized inverses applied to robotics. *Int. J. of Robotics Research*, 12(1):1–19, December 1993.

[Dune 10] C. Dune, A. Herdt, O. Stasse, P-B. Wieber, K. Yokoi, E. Yoshida. – Cancelling the sway motion of dynamic walking in visual servoing. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'10)*, Tapei, Taiwan, October 2010.

[Duong 12] D. Duong, J-P. Laumond, F. Lamiraux. – Experiments on whole-body manipulation and locomotion with footstep real-time optimization. December 2012.

[EAL 12] Engineered Arts Limited EAL. – Theater of robots. – http://www.theatreofrobots.co.uk/, 2012. (web address available in Jan 2013).

[El Khoury 13] A. El Khoury, F. Lamiraux, M. Taïx. – Optimal motion planning for humanoid robots. – *IEEE Int. Conf. on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, May 2013.

[English 99] C. English, D. Russell. – Implementation of variable joint stiffness through antagonistic actuation using rolamite springs. *Mechanism and Machine Theory*, 34(1):27–40, 1999.

[Enoch 12] A. Enoch, A. amd Sutas, S. Nakaoka, Vijayakumar S. – Blue: A bipedal robot with variable stiffness and damping. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'12)*, Osaka, Japan, November 2012.

[Escande 07] A. Escande, S. Miossec, A. Kheddar. – Continuous gradient proximity distances for humanoids free-collision optimized-postures generation. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'07)*, Pittsburgh, USA, November 2007.

[Escande 08a] A. Escande. – *Planification de points d'appui pour la gńération de mouvements acycliques : application aux humanoïdes.* – Evry, France, PhD. Thesis, Univ. Evry - Val d'Esonne, November 2008. A. Kheddar supervising.

[Escande 08b] A. Escande, A. Kheddar, S. Miossec, S. Garsault. – Planning support contact-points for acyclic motions and experiments on HRP-2. – *International Symposium on Experimental Robotics (ISER'08)*, Athens, Greece, July 2008.

[Escande 13] A. Escande, A. Kheddar, S. Miossec. – Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 2013. – [in press].

[Espiau 92] B. Espiau, F. Chaumette, P. Rives. – A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[Evrard 08] P. Evrard, F. Keith, J.-R. Chardonnet, A. Kheddar. – Framework for haptic interaction with virtual avatars. – *IEEE Int. Symp. on Robot and Human Interactive Communication*, Munich, Germany, August 2008.

[Featherstone 08] R. Featherstone. – *Rigid body dynamics algorithms.* – Springer, 2008.

[Flacco 12] F. Flacco, A. De Luca, I. Sardellitti, N. Tsagarakis. – On-line estimation of variable stiffness in flexible robot joints. *Int. Journal of Robotics Research*, 31(13):1556–1577, 2012.

[Foissotte 10] T. Foissotte, O. Stasse, P.-B. Wieber, A. Escande, A Kheddar. – Autonomous 3d object modeling by a humanoid using an optimization-driven next-best view formulation. *International Journal on Humanoid Robotics, Special issue on Cognitive Humanoid Vision*, 7(3):407–428, 2010.

[Fourquet 07] J-Y. Fourquet, V. Hue, P. Chiron. – Olarge: on kinematic schemes and regularization for automatic generation of human motion and ergonomic evaluation of workplaces. – *IEEE Industrial Electronics Society (IECON'07)*, 2007.

[Gans 07] N. Gans, S. Hutchinson. – Stable visual servoing through hybrid switched-system control. *IEEE Trans. on Robotics*, 23(3):530–540, June 2007.

[GarciaAracil 05] N. Garcia-Aracil, E. Malis, R. Aracil-Santonja, C. Perez-Vidal. – Continuous visual servoing despite the changes of visibility in image features. *IEEE Trans. on Robotics*, 21(6):415–421, April 2005.

[Gergondet 12] P. Gergondet, D. Petit, A. Kheddar. – Steering a robot with a brain-computer interface: impact of the video feedback on bci performance. – *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Paris, France, September 2012.

[Ghallab 04] M. Ghallab, D. Nau, P. Traverso. – *Automated Planning: Theory and Practice.* – Morgan Kauffmann Publishers, 2004.

[Gharbi 09] M. Gharbi, J. Cortés, T. Siméon. – Roadmap composition for multi-arm systems path planning. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, St Louis, USA, October 2009.

[Ginsberg 83] C. Ginsberg, D. Maxwell. – Graphical marionette. – *ACM SIGGRAPH/SIGART Workshop on Motion*, pp. 172–179, April 1983.

[Golub 96] G. Golub, C. Van Loan. – *Matrix computations.* – John Hopkins University Press, 3rd edition, 1996.

[Gravot 06] F. Gravot, A. Haneda, K. Okada, M. Inaba. – Cooking for humanoid robot, a task that needs symbolic and geometric reasonings. – *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*, Orlando, USA, May 2006.

[Grebenstein 11] M. Grebenstein, A. Albu-Schaffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, Haddadin et al. – The DLR hand arm system. – *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[Grizzle 01] J.W. Grizzle, G. Abba, F. Plestan. – Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Trans. on Robotics and Automation*, 46(1):51–64, January 2001.

[Guennebaud 10] G. Guennebaud, B. Jacob et al. – Eigen v3. – http://eigen.tuxfamily.org, 2010.

[Guizzo 10] E. Guizzo. – France developing advanced humanoid robot romeo. *IEEE Spectrum Automaton Blog*, December 13, 2010.

[Guizzo 11] E. Guizzo. – Fukushima robot operator writes tell-all blog. *IEEE Spectrum Automaton Blog*, August, 23, 2011.

[Guizzo 12] E. Guizzo, E. Ackerman. – DARPA robotics challenge: Here are the official details. *IEEE Spectrum Automaton Blog*, April, 10 2012.

[Habibi 00] S. Habibi, A. Goldenberg. – Design of a new high-performance electrohydraulic actuator. *IEEE/ASME Transactions on Mechatronics*, 5(2):158–164, June 2000.

[Hait 02] A. Hait, T. Siméon, M. Taïx. – Algorithms for rough terrain trajectory planning. *Advanced Robotics*, 16(8):673–699, 2002.

[Han 98] L. Han, J. Trinkle. – Dextrous manipulation by rolling and finger gaiting. – *IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, pp. 730–735, Leuven, Belgium, May 1998.

[Hartikainen 92] K. Hartikainen, A. Halme, H. Lehtinen, K. Koskinen. – MECANT I: a six legged walking machine for research purposes in outdoor environment. – *IEEE Int. Conf. on Robotics and Automation (ICRA'92)*, pp. 157–163, Nice, France, May 1992.

[Hauser 08] K. Hauser, T. Bretl, J-C. Latombe, K. Harada, B. Wilcox. – Motion planning for legged robots on varied terrain. *Int. Journal of Robotics Research*, 27(11-12):1325–1349, November 2008.

[Hayet 12] J-B. Hayet, C. Esteves, G. Arechavaleta, O. Stasse, E. Yoshida. – Humanoid locomotion planning for visually-guided tasks. *International Journal of Humanoid Robotics*, 9(2), 2012.

[Herdt 10] A. Herdt, H. Diedam, P-B. Wieber, D. Dimitrov, K. Mombaur, M. Diehl. – Online walking motion generation with automatic footstep placement. *Advanced Robotics*, 24(5-6):719–737, 2010.

[Hirai 98] K. Hirai, M. Hirose, Y. Haikawa, T. Takenaka. – The development of honda humanoid robot. – *IEEE Int. Conf. on Robotics and Automation (ICRA'98)*, Leuven, Belgium, May 1998.

[Hirukawa 06] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, M. Morisawa. – A universal stability criterion of the foot contact of legged robots-adios zmp. – *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*, Orlando, USA, May 2006.

[Hofmann 09] A. Hofmann, M. Popovic, H. Herr. – Exploiting angular momentum to enhance bipedal center-of-mass control. – *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan, 2009.

[Hogan 84] N. Hogan. – Impedance control: An approach to manipulation. – *American Control Conference (ACC'84)*, pp. 304–313, 1984.

[Hogan 85] N. Hogan. – Impedance control: An approach to manipulation: theory, implementation and applications (part 1 to 3). *Journal of Dynamic Systems, Measurement, and Control*, 107(1):1–24, March 1985.

[Hogan 88] N. Hogan. – On the stability of manipulators performing contact tasks. *Robotics and Automation, IEEE Journal of*, 4(6):677–686, 1988.

[Ierusalimschy 96] R. Ierusalimschy, L. De Figueiredo, W. Filho. – Lua-an extensible extension language. *Software Practice and Experience*, 26(6):635–652, 1996.

[Inamura 04] Tetsunari Inamura, Iwaki Toshima, Hiroaki Tanie, Yoshihiko Nakamura. – Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research*, 23(4-5):363–377, 2004.

[Ind 10] Kawada Ind. – Nextage: next generation of industrial robot. – http://nextage.kawada.jp/en, 2010. (web address available in Jan 2013).

[Inoue 74] H. Inoue. – Force feedback in precise assembly tasks. *MIT AI Memos*, August 1974.

[Isermann 82] H. Isermann. – Linear lexicographic optimization. *Operations Research Spektrum*, 4:223–228, 1982.

[JacquierBret 09] J. Jacquier-Bret, N. Rezzoug, P. Gorce. – Adaptation of joint flexibility during a reach-to-grasp movement. *Motor Control*, 13(3):342–361, July 2009.

[Jafari 13] A. Jafari, N. Tsagarakis, D. Caldwell. – A novel intrinsically energy efficient development of a novel actuator with adjustable stiffness (AwAS). *IEEE Transactions on Mechatronics*, 18(1), January 2013.

[Kaiser 12] P. Kaiser, D. Berenson, N. Vahrenkamp, T. Asfour, R. Dillmann, S. Srinivasa. – Constellation – an algorithm for finding robot configurations that satisfy multiple constraints. – *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, Saint Paul, USA, May 2012.

[Kajita 03] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, H. Hirukawa. – Biped walking pattern generation by using preview control of zero-moment point. – *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, September 2003.

[Kajita 10a] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, K. Yokoi. – Biped walking stabilization based on linear inverted pendulum tracking. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'10)*, Tapei, Taiwan, October 2010.

[Kajita 10b] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, K. Yokoi. – Biped walking stabilization based on linear inverted pendulum tracking. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'10)*, Tapei, Taiwan, October 2010.

[Kalakrishnan 11] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, Stefan Schaal. – Stomp: Stochastic trajectory optimization for motion planning. – *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[Kaneko 94] K. Kaneko, K. Komoriya, K. Ohnishi, K. Tanie. – Manipulator control based on a disturbance observer in the operational space. – *IEEE Int. Conf. on Robotics and Automation (ICRA'94)*, vol. 2, pp. 902–909, 1994.

[Kaneko 02] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, T. Isozumi. – Design of prototype humanoid robotics platform for hrp. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'00)*, Lausanne, Switzerland, October 2002.

[Kaneko 08] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, K. Akachi. – Humanoid robot hrp-3. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'08)*, Nice, France, September 2008.

[Kaneko 09] K. Kaneko, F. Kanehiro, M. Morisawa, K. Miura, S. Nakaoka, S. Kajita. – Cybernetic human hrp-4c. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'09)*, Paris, France, December 2009. – Best paper award.

[Kaneko 12] K. Kaneko, F. Kanehiro, M. Morisawa, E. Yoshida, J. Laumond. – Disturbance observer that estimates external force acting on humanoid robots. – *12th IEEE International Workshop on Advanced Motion Control (AMC)*, Sarajevo, Bosnia, March 2012.

[Kanoun 09a] O. Kanoun. – *Contribution à la plannification de mouvements pour robots humanoïdes (in English).* – France, PhD. Thesis, Univ. Toulouse, October 2009.

[Kanoun 09b] O. Kanoun, F. Lamiraux, F. Kanehiro, E. Yoshida, Laumond J-P. – Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. – *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan, May 2009.

[Kanoun 11a] O. Kanoun, F. Lamiraux, P-B. Wieber. – Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks. *IEEE Trans. on Robotics*, 27(4):785–792, August 2011.

[Kanoun 11b] Oussama Kanoun, Jean-Paul Laumond, Eiichi Yoshida. – Planning foot placements for a humanoid robot: A problem of inverse kinematics. *Int. Journal of Robotics Research*, 30(4):476–485, 2011.

[Kato 73] I. Kato. – Development of wabot-1. *Biomechanism*, 2:173–214, 1973.

[Kazerooni 08] H. Kazerooni. – Exoskeletons for human performance augmentation. *Handbook of Robotics*, B. Siciliano, O. Khatib Edts. – Springer-Verlag, 2008.

[Keith 11] F. Keith. – *Module logiciel de contrÃ´le dynamique de mouvement.* – Rapport de recherche, Grenoble, France, Inria Rhones Alpes, December 2011. (Romeo Project, Delivrable 7.3.1).

[Khatib 86] O. Khatib. – Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5(1):90–98, March 1986.

[Khatib 87] O. Khatib. – A unified approach for motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research*, 3(1):43–53, February 1987.

[Khatib 08] O. Khatib, P. Thaulad, T. Yoshikawa, J. Park. – Torque-position transformer for task control of position controlled robots. – *IEEE Int. Conf. on Robotics and Automation (ICRA'08)*, Passadena, USA, May 2008.

[Koga 94] Y. Koga, K. Kondo, J. Kuffner, J-C. Latombe. – Planning motions with intentions. – *Annual conference on Computer graphics and interactive techniques*, pp. 395–408. ACM, 1994.

[Koolen 12] Twan Koolen, Tomas De Boer, John Rebula, Ambarish Goswami, Jerry Pratt. – Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models. *Int. Journal of Robotics Research*, 31(9):1094–1113, 2012.

[Kuffner 00] J. Kuffner, S. LaValle. – RRT-connect: An efficient approach to single-query path planning. – *IEEE Int. Conf. on Robotics and Automation (ICRA'00)*, San Francisco, USA, April 2000.

[Kuffner 03] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, H. Inoue. – Motion planning for humanoids robots. – *International Symposium of Robotics Research (ISRR'03)*, Siena, Italia, October 2003.

[Lamon 04] P. Lamon, A. Krebs, M. Lauria, R. Siegwart, S. Shooter. – Wheel torque control for a rough terrain rover. – *IEEE Int. Conf. on Robotics and Automation (ICRA'04)*, New Orlean, Louisiane, April 2004.

[Latash 02] M. Latash, J. Scholz, G. Schöner. – Motor control strategies revealed in the structure of motor variability. *Exercise and sport sciences reviews*, 30(1):26–31, January 2002.

[Laumond 98] J.P. Laumond (Edts). – *Robot motion planning and control.* – Springer, vol. 229, 1998.

[Laumond 06] J.P. Laumond. – Motion planning for PLM : State of the art and perspectives. *Int. Journal on Product Lifecycle Management*, 1:129–142, 2006.

[Laumond 12] J-P. Laumond. – Le corps et sa structure. locomotion humaine et humanoïde : fondements calculatoires. – Collège de France, class of the Chaire d'Innovation technologique Liliane Bettencourt (2011-2012), March, 12, 2012. (video, in French).

[Laumond 13] J-P. Laumond. – *Simplexité et Complexité*, chap. Simplexité et Robotique: vers un génie de l'action encorporée. – Odile Jacob, J-L Petit, A. Berthoz Edts, 2013. [in French, to appear].

[Lawrence 97] C. Lawrence, J. L. Zhou, A. L. Tits. – *User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints.* – Rapport de Recherche n TR-94-16r1, College Park, USA, Institute for Systems Research, Univ Maryland, 1997.

[Lee 88] H.Y. Lee, C.G. Liang. – Displacement analysis of the general spatial 7-link 7r mechanism. *Mechanism and Machine Theory*, 23(3):219–226, 1988.

[Leineweber 03] D. Leineweber, I. Bauer, H. Bock, J. Schlöder. – An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization - part i: theoretical aspects. *Computers & Chemical Engineering*, 27(2):157–166, February 2003.

[Lengagne 13] S. Lengagne, J.Vaillant, A. Kheddar, E. Yoshida. – Generation of whole-body optimal dynamic multi-contact motions. *Int. Journal of Robotics Research*, 2013. – [in press].

[Li 08] H. Li, B. Williams. – Generative planning for hybrid systems based on flow tubes. *International Conference on Automated Planning and Scheduling (ICAPS'08)*, September 2008.

[Li 12] Z. Li, B. Vanderborght, N. Tsagarakis, L. Colasanto, D. Caldwell. – Stabilization for the compliant humanoid robot coman exploiting intrinsic and controlled compliance. – *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, Saint Paul, USA, May 2012.

[Liégeois 77] A. Liégeois. – Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man and Cybernetics*, 7(12):868–871, December 1977.

[Likhachev 09] M. Likhachev, B. Marthi, C. McGann, D. E. Smith (Edts). – *First Workshop on Bridging The Gap Between Task And Motion Planning*, AAAI International Conference on Automated Planning and Scheduling (ICAPS'09). – Thessaloniki, Greece, September 2009.

[Liu 11] M. Liu, A. Micaelli, P. Evrard, A. Escande, C. Andriot. – Interactive dynamics and balance of a virtual character during manipulation tasks. – *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[LozanoPerez 83] Tomas Lozano-Perez. – Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 100(2):108–120, 1983.

[Maeda 01] Y. Maeda, T. Hara, T. Arai. – Human-robot cooperative manipulation with motion estimation. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'01)*, Maui, Hawai, October 2001.

[Maes 10] C. Maes. – *A Regularized Active-Set Method for Sparse Convex Quadratic Programming.* – USA, PhD. Thesis, Institute for Computational and Mathematical Engineering, Stanford University, November 2010.

[Malti 11] A. Malti, M. Taïx, F. Lamiraux. – A general framework for planning landmark-based motions for mobile robots. *Advanced Robotics*, 25(11):1427–1450, 2011.

[Manocha 94] D. Manocha, J.F. Canny. – Efficient inverse kinematics for general 6r manipulators. *IEEE Trans. on Robotics and Automation*, 10(5):648–657, 1994.

[Marani 02] G. Marani, K. Jinhyun, Y. Junku, K. Wan. – A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators. – *IEEE Int. Conf. on Robotics and Automation (ICRA'02)*, pp. 1973–1978, Washingtown DC, USA, May 2002.

[Marchand 98] E. Marchand, G. Hager. – Dynamic sensor planning in visual servoing. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98)*, Leuven, Belgium, May 1998.

[Marchand 05] E. Marchand, F. Spindler, F. Chaumette. – Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, December 2005.

[MarinUrias 09] L. Marin-Urias, E. Sisbot, A. Pandey, R. Tadakuma, R. Alami. – Towards shared attention through geometric reasoning for human robot interaction. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'09)*, Paris, France, December 2009.

[Martinelli 12] A. Martinelli. – Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Trans. on Robotics*, 28(1):44–60, 2012.

[Mellor 11] M. Mellor. – Fighting robots. – https://vimeo.com/18530627, 2011. (Vimeo video).

[Mezouar 02] Y. Mezouar, F. Chaumette. – Path planning for robust image-based control. *IEEE Trans. on Robotics and Automation*, 18(4):534–549, August 2002.

[Michel 05] P. Michel, J. Chestnutt, J. Kuffner, T. Kanade. – Vision-guided humanoid footstep planning for dynamic environments. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'05)*, pp. pages 13–18, Tsukuba, Japan, December 2005.

[Mittendorfer 11] P. Mittendorfer, G. Cheng. – Humanoid multimodal tactile-sensing modules. *IEEE Trans. on Robotics*, 27(3):401–410, June 2011.

[Mittendorfer 13] P. Mittendorfer, E. Yoshida, T. Moulard, G. Cheng. – A general tactile approach for grasping unknown objects with a humanoid robot. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'13)*, Tokyo, Japan, November 2013. – [submitted].

[Mizumoto 09] T. Mizumoto, H. Tsujino, T. Takahashi, T. Ogata, H. Okuno. – Thereminist robot: development of a robot theremin player with feedforward and feedback arm control based on a theremin's pitch model. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, St Louis, USA, October 2009.

[Morales 06] A. Morales, T. Asfour, P. Azad, S. Knoop, R. Dillmann. – Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, Beijin, China, October 2006.

[Mordatch 12] I. Mordatch, E. Todorov, Z. Popović. – Discovery of complex behaviors through contact-invariant optimization. *ACM SIGGRAPH'12*, 31(4):43, 2012.

[Morisawa 07] M. Morisawa, K. Harada, S. Kajita, S. Nakaoka, K. Fujiwara, F. Kanehiro, K. Kaneko, H. Hirukawa. – Experimentation of humanoid walking allowing immediate modification of foot place based on analytical solution. – *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italia, April 2007.

[Moulard 12a] T. Moulard. – *Numerical Optimization for robotics and closed-loop trajectory execution.* – Toulouse, France, PhD. Thesis, Univ. Toulouse, September 2012.

[Moulard 12b] T. Moulard. – SOT: redundant manipulator control. – Stack Robot Operating System (ROS.org), 2012. http://www.ros.org/wiki/redundant_manipulator_control.

[Moulard 12c] T. Moulard, P.F. Alcantarilla, F. Lamiraux, O. Stasse, F. Dellaert. – Reliable indoor navigation on humanoid robots using vision-based localization. – *Submitted to ICRA'13 (to be resubmitted)*, 2012.

[Moulard 12d] T. Moulard, F. Lamiraux, O. Stasse. – Trajectory following for legged robots. – *IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012. – www.youtube.com/watch?v=cUZ0nNiPs70.

[Myers 99] K. Myers. – Cpef : A continuous planning and execution framework. *AI Magazine*, 20(4):63–69, December 1999.

[Nabeshima 12] C. Nabeshima, H. Kawamoto, Y. Sankai. – Strength testing machines for wearable walking assistant robots based on risk assessment of robot suit hal. – *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, Saint Paul, USA, May 2012.

[Nakamura 05] Y. Nakamura, K. Yamane, Y. Fujita, I. Suzuki. – Somatosensory computation for man-machine interface from motion-capture data and musculoskeletal human model. *IEEE Trans. on Robotics*, 21(1):58–66, 2005.

[Nakaoka 07] Shin'ichiro Nakaoka, Atsushi Nakazawa, Fumio Kanehiro, Kenji Kaneko, Mitsuharu Morisawa, Hirohisa Hirukawa, Katsushi Ikeuchi. – Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *The International Journal of Robotics Research*, 26(8):829–844, 2007.

[Nakaoka 10] S. Nakaoka, S. Kajita, K. Yokoi. – Intuitive and flexible user interface for creating whole body motions of biped humanoid robots. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'10)*, Tapei, Taiwan, October 2010. – Best Paper Finalist, NTF Award Finalist for Entertainment Robots and Systems.

[Nelson 95] B. Nelson, P. Khosla. – Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limits avoidance. *Int. Journal of Robotics Research*, 14(3):255–269, June 1995.

[Nelson 12] G. Nelson, A. Saunders, N. Neville, B. Swilling, J. Bondaryk, D. Billings, C. Lee, R. Playter, M. Raibert. – Petman: A humanoid robot for testing chemical protective clothing. *Journal of the Robotics Society of Japan*, 30(4):372–377, 2012.

[Nemirovski 94] A. Nemirovski. – *Efficient methods in convex programming.* – Technion, the Israel institute of technology, 1994. Fall Semester 1994/95.

[Nenchev 89] D.N. Nenchev. – Redundancy resolution through local optimization: A review. *Journal of Robotic Systems*, 6(6):769–798, 1989.

[Nishiwaki 07] K. Nishiwaki, J. Kuffner, S. Kagami, M. Inaba, H. Inoue. – The experimental humanoid robot h7: a research platform for autonomous behaviour. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):79–107, 2007.

[Ogura 06] Y. Ogura, H. Aikawa, K. Shimomura, A. Morishima, H. Lim, A. Takanishi. – Development of a new humanoid robot wabian-2. – *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*, Orlando, USA, May 2006.

[Olaru 09] I. Olaru, S. Krut, F. Pierrot. – Novel mechanical design of biped robot sherpa using 2 dof cable differential modular joints. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, St Louis, USA, October 2009.

[Ott 08] C. Ott, A. Kugi, Y. Nakamura. – Resolving the problem of non-integrability of nullspace velocities for compliance control of redundant manipulators by using semi-definite lyapunov functions. – *IEEE Int. Conf. on Robotics and Automation (ICRA'08)*, Passadena, USA, May 2008.

[Ott 10] C. Ott, C. Baumgartner, J. Mayr, M. Fuchs, R. Burger, D. Lee, O. Eiberger, A. Albu-Schaffer, M. Grebenstein, G. Hirzinger. – Development of a biped robot with torque controlled joints. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'10)*, Nashville, USA, October 2010.

[Pang 00] J. Pang, J. Trinkle. – Stability characterizations of rigid body contact problems with coulomb friction. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 80(10):643–663, 2000.

[Park 06a] J. Park. – *Control Strategies for Robots in Contact.* – USA, PhD. Thesis, Stanford University, March 2006.

[Park 06b] J. Park, O. Khatib. – A haptic teleoperation approach based on contact force control. *Int. Journal of Robotics Research*, 25(5):575–591, 2006.

[Peinado 05] M. Peinado, R. Boulic, B. Le Callennec, D. Meziat. – Progressive cartesian inequality constraints for the inverse kinematics control of articulated chains. – *Eurographics'05*, Dublin, Ireland, September 2005.

[Perrin 10] N. Perrin, O. Stasse, F. Lamiraux, E. Yoshida. – Approximation of feasibility tests for reactive walk on hrp-2. – *IEEE International Conference on Robotics and Automation (ICRA'10)*, Anchorage, USA, May 2010.

[Petrovskaya 07] A. Petrovskaya, J. Park, O. Khatib. – Probabilistic estimation of whole body contacts for multi-contact robot control. – *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italy, April 2007.

[Pettré 08] J. Pettré, M. Kallmann, M. Lin. – Motion planning and autonomy for virtual humans. – *ACM SIGGRAPH 2008 classes (SIGGRAPH '08)*, p. 42, 2008.

[Pham 12] Q.C. Pham, Y. Nakamura. – Affine trajectory deformation for redundant manipulators. – *Robotics: Science and Systems (RSS'12)*, Sydney, Australia, July 2012. – Best Paper Award.

[Pieper 68] D. Pieper. – *The Kinematics of Manipulators under Computer Control.* – PhD. Thesis, Stanford University, 1968.

[Pongas 07] D. Pongas, M. Mistry, S. Schaal. – A robust quadruped walking gait for traversing rough terrain. – *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italia, April 2007.

[Pratt 95] G. Pratt, M. Williamson. – Series elastic actuators. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'95)*, pp. 399–406, 1995.

[Pratt 06] Jerry Pratt, John Carff, Sergey Drakunov, Ambarish Goswami. – Capture point: A step toward humanoid push recovery. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'06)*, Genova, Italy, November 2006.

[Pratt 12] Jerry Pratt, Twan Koolen, Tomas De Boer, John Rebula, Sebastien Cotton, John Carff, Matthew Johnson, Peter Neuhaus. – Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid. *Int. Journal of Robotics Research*, 31(10):1117–1133, 2012.

[Primrose 86] E. Primrose. – On the input-output equation of the general 7r-mechanism. *Mechanism and Machine Theory*, 21(6):509–510, 1986.

[Quigley 09] M. Quigley, K. Conley, B. . Gerkey P, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng. – Ros: an open-source robot operating system. – *ICRA Workshop on Open Source Software*, 2009.

[Raibert 84] M. Raibert, H. Brown, M. Chepponis. – Experiments in balance with a 3d one-legged hopping machine. *Int. Journal of Robotics Research*, 3(2):75–92, 1984.

[Raibert 08] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, the Bigdog Team. – Bigdog, the rough-terrain quadruped robot. – *IFAC, 17th World Congress*, Seoul, Korea, June 2008.

[Rasmussen 01] J. Rasmussen, M. Damsgaard, M. Voigt. – Muscle recruitment by the min/max criterion – a comparative numerical study. *Journal of Biomechanics*, 34(3):409–415, 2001.

[Ratliff 09] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, Siddhartha Srinivasa. – CHOMP: Gradient optimization techniques for efficient motion planning. – *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan, May 2009.

[Raunhardt 07] D. Raunhardt, R. Boulic. – Progressive clamping. – *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italy, April 2007.

[Rusu 09] R. Rusu, I. Sucan, B. Gerkey, S. Chitta, M. Beetz, L. Kavraki. – Real-time perception-guided motion planning for a personal robot. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, St Louis, USA, October 2009.

[Saidi 07] F. Saidi, O. Stasse, K. Yokoi, F. Kanehiro. – Online object search with a humanoid robot. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'07)*, San Diego, USA, november 2007.

[Sakamoto 09] D. Sakamoto, H. Ishiguro. – Geminoid: Remote-controlled android system for studying human presence. *Kansei Engineering International*, 8(1):3–9, 2009.

[Salini 09] J. Salini, S. Barthélemy, P. Bidaud. – Lqp controller design for generic whole body motion. – *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, Kobe, Japan, May 2009.

[Samson 91] C. Samson, M. Le Borgne, B. Espiau. – *Robot Control: the Task Function Approach.* – Clarendon Press, Oxford, United Kingdom, 1991.

[Sardain 04] P. Sardain, G. Bessonnet. – Forces acting on a biped robot. center of pressure-zero moment point. *IEEE Trans. on System, Man and Cybernetics (Part A: Systems and Humans)*, 34(5):630–637, September 2004.

[Sardellitti 12] I. Sardellitti, G. Medrano-Cerda, N. Tsagarakis, A. Jafari, D. Caldwell. – A position and stiffness control strategy for variable stiffness actuators. – *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, Saint Paul, USA, May 2012.

[Schultz 10] G. Schultz, K. Mombaur. – Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, 15(5):783–792, October 2010.

[Segvic 09] S. Segvic, A. Remazeilles, A. Diosi, F. Chaumette. – A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2):172–187, February 2009.

[Sentis 07] L. Sentis. – *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. – USA, PhD. Thesis, Stanford University, July 2007.

[Shin 10] D. Shin, I. Sardellitti, Y.L. Park, O. Khatib, M. Cutkosky. – Design and control of a bio-inspired human-friendly robot. *Int. Journal of Robotics Research*, 29(5):571–584, 2010.

[Sian 05] N. Sian, K. Yokoi, S. Kajita, F. Kanehiro, K Tanie. – A switching command-based whole-body operation method for humanoid robots. *IEEE/ASME Transactions on Mechatronics*, 10(5):546–559, October 2005.

[Siciliano 90] B. Siciliano. – Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent & Robotic Systems*, 3(3):201–212, 1990.

[Siciliano 91] B. Siciliano, J-J. Slotine. – A general framework for managing multiple tasks in highly redundant robotic systems. – *IEEE Int. Conf. on Advanced Robotics (ICAR'91)*, Pisa, Italy, June 1991.

[Siegwart 02] R. Siegwart, P. Lamon, T. Estier, M. Lauria, R. Piguet. – Innovative design for wheeled locomotion in rough terrain. *Robotics and Autonomous systems*, 40(2):151–162, 2002.

[Siméon 04] T. Siméon, J-P. Laumond, J. Cortés, A. Sahbani. – Manipulation planning with probabilistic roadmaps. *Int. Journal of Robotics Research*, 23(7-8):729–746, August 2004.

[Smith 00] D. Smith, J. Frank, A. Jónsson. – Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15(1):47–83, March 2000.

[Solveig 12] M. Solveig. – Danse with robots. – http://en.futuroscope.com/attractions-and-shows/thrills/dances-with-robots, 2012. (web address available in Jan 2013).

[Soueres 03] P. Soueres, V. Cadenat, M. Djeddou. – Dynamical sequence of multi-sensor based tasks for mobile robots navigation. – *7th Symp. on Robot Control (SYROCO'03)*, pp. 423–428, Wroclaw, Poland, September 2003.

[Stasse 09] O. Stasse, N. Perrin, P.-B. Wieber, N. Mansard, F. Lamiraux. – Very fast decision making for whole body motion generation with humanoid robots. – *IEEE RO-MAN Workshop on Human-Synergy*, 2009.

[Stilman 10] Mike Stilman, Jon Olson, Will Gloss. – Golem krang: Dynamically stable humanoid robot for mobile manipulation. – *IEEE Int. Conf. on Robotics and Automation (ICRA'10)*, Anchorage, USA, May 2010.

[Stoer 02] J. Stoer, R. Bulirsch. – *Introduction to Numerical Analysis*, chap. 7.3: Boundary-value problem. – Springer-Verlag, ISBN 978-0-387-95452-3, 2002, 3rd edition.

[Strickland 12] E. Strickland. – Good-bye, wheelchair. *IEEE Spectrum*, 49(1):30–32, 2012.

[Sturman 94] D. Sturman. – A brief history of motion capture for computer character animation. *ACM SIGGRAPH'94, Character Motion Systems (Course notes)*, April 1994.

[Sugihara 02] T. Sugihara, Y. Nakamura. – Whole-body cooperative balancing of humanoid robot using COG jacobian. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02)*, Lausanne, Switzerland, October 2002.

[Sugihara 11] T. Sugihara. – Solvability-unconcerned inverse kinematics by the levenberg-marquardt method. *IEEE Trans. on Robotics*, 27(5):984–991, October 2011.

[Sung 96] Y. Sung, D. Cho, M. Chung. – A constrained optimization approach to resolving manipulator redundancy. *Journal of robotic systems*, 13(5):275–288, December 1996.

[Taïx 12] M. Taïx, C. Briand, P. Truillet, A. de Bonneval, I. Ferrane, J. Pinquier. – Nao : Support fondamental à la pédagogie par projets en master 2. – *NAO Tech Day*, Paris, France, June 2012.

[Takahashi 05] Y. Takahashi, K. Nishiwaki, S. Kagami, H. Mizoguchi, H. Inoue. – High-speed pressure sensor grid for humanoid robot foot. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'05)*, Edmonton, Canada, August 2005.

[Tassa 12] Y. Tassa, T. Erez, E. Todorov. – Synthesis and stabilization of complex behaviors through online trajectory optimization. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'12)*, Lisbon, Portugal, 2012.

[Tassa 13] Y. Tassa, T. Wu, J. Movellan, E. Todorov. – Modeling and identification of pneumatic actuators. – *IEEE Int. Conf. on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, May 2013. – [submitted].

[Tolani 00] D. Tolani, A. Goswami, N.I. Badler. – Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

[Tonietti 05] G. Tonietti, R. Schiavi, A. Bicchi. – Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction. – *IEEE Int. Conf. on Robotics and Automation (ICRA'05)*, Barcelona, Spain, April 2005.

[Townsend 99] W. Townsend, J. Guertin. – Teleoperator slave-wam design methodology. *Industrial Robot: An International Journal*, 26(3):167–177, 1999.

[Tsagarakis 11] N. Tsagarakis, Z. Li, J. Saglia, D. Caldwell. – The design of the lower body of the compliant humanoid robot iCub. – *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[Vahrenkamp 09] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, R. Dillmann. – Humanoid motion planning for dual-arm manipulation and re-grasping tasks. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, St Louis, USA, October 2009.

[Vanderborght 10] B. Vanderborght. – *Dynamic stabilisation of the biped Lucy powered by actuators with controllable stiffness.* – Springer, vol. 63 of *Springer tracts in advanced robotics (STAR)*, 2010.

[VanLoan 85] C. Van Loan. – On the method of weighting for equality-constrained least-squares problems. *SIAM Journal on Numerical Analysis*, 22(5):851–864, 1985.

[Visser 10] L. Visser, R. Carloni, R. Unal, S. Stramigioli. – Modeling and design of energy efficient variable stiffness actuators. – *IEEE Int. Conf. on Robotics and Automation (ICRA'10)*, Anchorage, USA, May 2010.

[Waldron 08] K. Waldron, J. Schmiedeler. – *Handbook of Robotics*, chap. 1: Kinematics. – Springer-Verlag, B. Siciliano and O. Khatib Edts, 2008, 1st edition.

[Walters 08] M. Walters, D. Syrdal, K. Dautenhahn, R. te Boekhorst, K. Koay. – Avoiding the uncanny valley: robot appearance, personality and consistency of behavior in an attention-seeking home scenario for a robot companion. *Autonomous Robots*, 24:159–178, 2008.

[Wampler 86] C. Wampler. – Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):93–101, January 1986.

[Wensing 13] P. Wensing, D. Orin. – Generation of dynamic humanoid behaviors through task-space control with conic optimization. – *IEEE Int. Conf. on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, May 2013.

[Weyrich 99] M. Weyrich, P. Drews. – An interactive environment for virtual manufacturing: the virtual workbench. *Computers in industry*, 38(1):5–15, 1999.

[Whitney 69] D. Whitney. – Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10(2):47–53, 1969.

[Whitney 76] D.E. Whitney. – Force feedback control of manipulator fine motions. *Productivity*, pp. 687–693, 1976.

[Wieber 02] Pierre-Brice Wieber. – On the stability of walking systems. – *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba, Japan, 2002.

[Wieber 08] P-B Wieber. – Viability and predictive control for safe locomotion. – *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'08)*, Nice, France, September 2008.

[Wieber 13] P-B. Wieber, R. Tedrake. – *Handbook of Robotics*, chap. Modeling and Control of Legged Robots. – Springer-Verlag, B. Siciliano and O. Khatib Edts, 2013, 2de edition. [preliminary version, to be published].

[Wisse 13] M. Wisse. – Factory in a day. – FP7 Call 9, Factories of the Future, October 2013.

[Wyrobek 08] K.A. Wyrobek, E.H. Berger, H.F.M. Van der Loos, J.K. Salisbury. – Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. – *IEEE Int. Conf. on Robotics and Automation (ICRA'08)*, Passadena, USA, May 2008.

[Yang 04] J. Yang, T. Marler, H. Kim, J. Arora, K. Abdel-Malek. – Multi-objective optimization for upper body posture prediction. – *AIAA/ISSMO multidisciplinary analysis and optimization conference*, vol. 319, p. 353, 2004.

[Yoshida 05] E. Yoshida, I. Belousov, C. Esteves, J-P. Laumond. – Humanoid motion planning for dynamic tasks. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'05)*, Tsukuba, Japan, november 2005.

[Yoshida 06] E. Yoshida, O. Kanoun, C. Esteves, J-P. Laumond. – Task-driven support polygon reshaping for humanoids. – *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'06)*, Genova, Italy, November 2006.

[Yoshikawa 84] T. Yoshikawa. – Analysis and control of robot manipulators with redundancy. – *First International Symposium on Robotics Research (ISRR'84)*, pp. 735–747. Mit Press Cambridge, MA, 1984.

[Zanchettin 12] A. Zanchettin, P. Rocco. – A general user-oriented framework for holonomic redundancy resolution in robotic manipulators using task augmentation. *IEEE Trans. on Robotics*, 28(2):514–521, April 2012.

[Zecca 09] M. Zecca, Y. Mizoguchi, K. Endo, F. Iida, Y. Kawabata, N. Endo, K. Itoh, A. Takanishi. – Whole body emotion expressions for kobian

humanoid robot-preliminary experiments with different emotional patterns. – *International Symposium on Robot and Human Interactive Communication (RO-MAN'09)*, Toyama, Japan, 2009.

[Zoss 06] A. Zoss, H. Kazerooni, A. Chu. – Biomechanical design of the berkeley lower extremity exoskeleton (bleex). *IEEE/ASME Transactions on Mechatronics*, 11(2):128–138, 2006.

Publications

Journals

[1] J-P. Laumond, N. Mansard, and J-B. Lassère. Optimality of robot motion. *Communications of the ACM*, April 2013. [submitted].

[2] O. Ramos, N. Mansard, O. Stasse, S. Hak, L. Saab, and C. Benazeth. Dynamic whole body motion generation for the dance of a humanoid robot. *IEEE Robotics and Automation Magazine*, 2013. [submitted].

[3] O. Stasse, P. Evrard, N. Mansard, P. Gergondet, A. Kheddar, K. Yokoi, T. Schauß, C. Passenberg, A. Peer, M. Buss, A. Weiss, M. Tscheligi, E. Gribovskaya, A. Billard, L. Blasi, J. Gancet, and M.J. Segarra-Martìnez. Multi-modal collaborative work with humanoid robots: a case study with HRP-2. *Advanced robotics*, 2013. [to be submitted soon].

[4] A. Escande, N. Mansard, and P-B. Wieber. Hierarchical quadratic programming. *Int. Journal of Robotics Research*, October 2012. [submitted].

[5] O. Ramos, N. Mansard, O. Stasse, and P. Souères. An advanced robotics motion generation framework for inferring the organization of human movements. *Computer Methods in Biomechanics and Biomedical Engineering*, 16(1), September 2013. [in press].

[6] L. Saab, O. Ramos, N. Mansard, P. Souères, and J-Y. Fourquet. Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Trans. on Robotics*, (2):346–362, April 2013.

[7] L. Saab, P. Souères, N. Mansard, and J-Y. Fourquet. Generation of human-like motion on anthropomorphic systems using inverse dynamics. *Computer Methods in Biomechanics and Biomedical Engineering*, 15(1):156–158, September 2012.

[8] J. Lee, N. Mansard, and J. Park. Intermediate desired value approach for task transition of robots in kinematic control. *IEEE Transaction on Robotics*, 28(6):1260 – 1277, December 2012.

[9] S. Hak, N. Mansard, O. Stasse, and J-P. Laumond. Reverse control for humanoid robot task recognition. *IEEE Trans. Sys. Man Cybernecics*, (6):1524–1537, December 2012.

[10] N. Mansard and F. Chaumette. Directional redundancy. *IEEE Trans. on Automatic Control*, 54(6):1179–1192, June 2009.

[11] N. Mansard, A. Remazeilles, and F. Chaumette. Continuity of varying-feature-set control laws. *IEEE Trans. on Automatic Control*, 54(11):2493 – 2505, November 2009.

[12] N. Mansard, O. Khatib, and A. Kheddar. A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Trans. on Robotics*, 25(3), June 2009.

[13] O. Stasse, B. Verrelst, A. Davison, N. Mansard, F. Saidi, B. Vanderborght, C. Esteves, and K. Yokoi. Integrating walking and vision to increase humanoid autonomy. *Int. Journal of Humanoid Robotics*, 5(9):287–310, June 2008.

[14] N. Mansard and F. Chaumette. Task sequencing for sensor-based control. *IEEE Trans. on Robotics*, 23(1):60 – 72, February 2007.

International Conferences

[15] K. Koch, K. Mombaur, and N. Mansard. Towards dynamical stepping over large obstacles for humanoid robots – a whole body optimal control approach. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'13)*, Tokyo, Japan, November 2013. [submitted].

[16] O. Ramos, N. Mansard, O. Stasse, and P. Souères. Walking on non-planar surfaces using an inverse dynamic stack of tasks. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'12)*, Osaka, Japan, November 2012.

[17] N. Mansard. A dedicated solver for fast operational-space inverse dynamics. In *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, St Paul, USA, may 2012.

[18] L. Saab, N. Mansard, P. Souères, J.Y. Fourquet, M. Sreenivasa, and Y. Nakamura. Whole-body torques for generating complex movements in humans and humanoids. In *11th Symp. on Robot Control (SYROCO'12)*, Dubrovnik, Croatia, September 2012.

[19] O. Ramos, L. Saab, S. Hak, and N. Mansard. Dynamic motion capture and edition using a stack of tasks. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'11)*, Bled, Slovenia, October 2011.

[20] C. Roussillon, A. Gonzalez, J. Solà, J-M. Codol, N. Mansard, S. Lacroix, and M. Devy. RT-SLAM: a generic and real-time visual SLAM implementation. In *Int. Conf. on Computer Vision Systems (ICVS)*, Sophia Antipolis, France, September 2011. Lecture notes in computer science 6962 (2011), pp. 31-40.

[21] L. Saab, O.E. Ramos, N. Mansard, J-Y. Fourquet, and P. Souères. Generic dynamic motion generation with multiple unilateral constraints. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'11)*, San Fransisco, USA, September 2011.

[22] F. Keith, P.B. Wieber, N. Mansard, and A. Kheddar. Analysis of the discontinuities in prioritized tasks-space control under discreet task scheduling operations. In *IEEE/RSJ*

Int. Conf. on Intelligent Robots and Systems (IROS'11), San Fransisco, USA, September 2011.

[23] N. Mansard. Vision-based motion generation and recognition for humanoid robots. In *European Conference on Eye Movements*, Marseille, France, August 2011.

[24] L. Saab, N. Mansard, F. Keith, J-Y. Fourquet, and P. Souères. Generation of dynamic motion for anthropomorphic system under prioritized equality and inequality constraints. In *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[25] J. Park. J. Lee, N. Mansard. Intermediate desired value approach for continuous transition among multiple tasks of robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, Shangai, China, May 2011.

[26] Mitsuharu Morisawa, Fumio Kanehiro, Kenji Kaneko, Nicolas Mansard, Joan Solà, Eiichi Yoshida, Kazuhito Yokoi, and Jean Paul Laumond. Combining suppression of the disturbance and reactive stepping for recovering balance. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'10)*, Taipei, Taiwan, October 2010.

[27] S. Hak, N. Mansard, and O. Stasse. Disambiguation of similar looking motions on an HRP-2 robot. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'10)*, Nashville, USA, December 2010.

[28] A. Escande, N. Mansard, and P-B. Wieber. A dedicated quadradic program for fast hierarchical-inverse-kinematic resolution. In *IEEE Int. Conf. on Robotics and Automation (ICRA'10)*, Anchorage, USA, May 2010.

[29] M. Morisawa, K. Harada, S. Kajita, K. Kaneko, J. Solà, E. Yoshida, N. Mansard, K. Yokoi, , and J-P. Laumond. Reactive stepping to prevent falling for humanoids. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'09)*, Paris, France, December 2009.

[30] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar. A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In *IEEE International Conference on Advanced Robotics (ICAR'09)*, Munich, Germany, June 2009.

[31] F. Keith, N. Mansard, S. Miossec, and A. Kheddar. Optimized time-warping tasks scheduling for smooth sequencing. In *10th Symp. on Robot Control (SYROCO'09)*, Gifu, Japan, September 2009.

[32] F. Keith, N. Mansard, S. Miossec, and A. Kheddar. From discrete mission schedule to continuous implicit trajectory using optimal time warping. In *19th International Conference on Automated Planning and Scheduling, ICAPS'09*, Thessaloniki, Greece, September 2009.

[33] P. Evrard, N. Mansard, O. Stasse, A. Kheddar, T. Schauss, C. Weber, A. Peer, and M. Buss. Intercontinental, multimodal, wide-range tele-cooperation using a humanoid robot. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, St Louis, USA, October 2009.

[34] F. Keith, N. Mansard, S. Miossec, and A. Kheddar. Optimization of tasks warping and scheduling for smooth sequencing of robotic actions. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, St Louis, USA, October 2009.

[35] O. Stasse, P. Evrard, N. Perrin, N. Mansard, and A. Kheddar. Fast foot prints replanning and motion generation during walking in physical human-humanoid interaction. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoid'09)*, Paris, France, December 2009.

[36] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. et Kheddar. Real-time (self)-collision avoidance task on a HRP-2 humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA'08)*, Las Passadenas, USA, May 2008.

[37] N. Mansard and O. Khatib. Continuous control law from unilateral constraints. In *IEEE Int. Conf. on Robotics and Automation (ICRA'08)*, Las Passadenas, USA, May 2008.

[38] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi. Visually-guided grasping while walking on a humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italia, April 2007.

[39] N. Mansard, M. Lopes, J. Santos-Victor, and F. Chaumette. Jacobian learning methods for tasks sequencing in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, Beijing, China, September 2006.

[40] A. Remazeilles, N. Mansard, and F. Chaumette. A qualitative visual servoing to ensure the visibility. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, Beijing, China, September 2006.

[41] N. Mansard and F. Chaumette. A new redundancy formalism for avoidance in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'05)*, Edmonton, Canada, August 2005.

[42] N. Mansard and F. Chaumette. Directional redundancy: a new approach of the redundancy formalism. In *IEEE Conf. on Decision and Control and European Control Conference, CDC/ECC 2005*, Sevilla, Spain, December 2005.

[43] N. Mansard and F. Chaumette. Visual servoing sequencing able to avoid obstacles. In *IEEE Int. Conf. on Robotics and Automation (ICRA'05)*, Barcelona, Spain, April 2005.

[44] N. Mansard, O. Aycard, and C. Koike. Hierarchy of behaviours. In *IEEE Int. Conf. on Robotics and Biomimetics, ROBIO'05*, Hong-Kong, China, July 2005.

[45] N. Mansard and F. Chaumette. Tasks sequencing for visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, Sendai, Japan, September 2004.

National Conferences and Workshops

[46] O. E. Ramos, M. García, N. Mansard, O. Stasse, J-B. Hayet, and P. Souères. Towards reactive vision-guided walking on rough terrain: an inverse-dynamics based approach. In M. Bennewitz and O. Stasse, editors, *Workshop on visual navigation for humanoid robots (IEEE ICRA'02)*, Karlsruhe, Germany, May 2013.

[47] S. Hak, N. Mansard, O. Ramos, L. Saab, O. Stasse, et al. Capture, recognition and imitation of anthropomorphic motion. In *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*, St Paul, USA, May 2012. (video session).

[48] N. Mansard. Contrôle, planification, reconnaissance de mouvement : la fonction de tâche, couteau suisse de la robotique humanoïde. In *Journées Nationales de la Recherche en Robotique*, La Rochelle, France, October 2011. Invited speaker.

[49] O. Stasse, F. Lamiraux, N. Mansard, P-B. Wieber, C. Dune, N. Perrin, A. Herdt, and S. Hak. R-blink : Génération très rapide de mouvements corps complet pour les robots humanoïdes. In *Journées Nationales de la Robotique Humanoïde*, Toulouse, France, April 2011.

[50] L. Saab, N. Mansard, F. Keith, P. Souères, and J-Y. Fourquet. Génération dynamique de mouvement corps-complet par optimisation de tâches hiérarchisées unilatérales et bilatérales. In *Journées Nationales de la Robotique Humanoïde*, Toulouse, France, April 2011.

[51] F. Lamiraux, N. Mansard, O. Stasse, and P-B. Wieber. R-blink: Génération très rapide de mouvements corps complet pour les robots humanoïdes. In *Journées Nationales de la Robotique Humanoïde*, Poitier, France, June 2010.

[52] S. Hak, N. Mansard, and O. Stasse. Identification de tâches pour le contrôle d'un robot humanoïde. In *Journées Nationales de la Robotique Humanoïde*, Poitier, France, June 2010.

[53] A. Abou Moughlbay, P. Martinet, and N. Mansard. Commande par vision d'un robot redondant multi-bras : Manipulation à deux bras avec le HRP2. In *Journées Nationales de la Robotique Humanoïde*, Poitier, France, June 2010.

[54] A. Kheddar, O. Stasse, P. Evrard, N. Mansard, E. Yo shida, and K. Yokoi. Démonstrateur humanoïde du projet robot@cwe. In *Journées Nationales de la Robotique Humanoïde*, Poitier, France, June 2010.

[55] A. Escande, P. Evrard, A. Kheddar, N. Mansard, S. Miossec, O. Stasse, and K. Yokoi. Travaux en Évitement de collision au jrl-japon. In *3emes Journées Nationales de la Robotique Humanoïde, JNRH'08*, Paris, France, May 2008.

[56] O. Stasse, B. Verrelst, A.J. Davison, N. Mansard, B. Vanderborght, C. Esteves, F. Saidi, and K. Yokoi. Integrating vision and walking to increase humanoid autonomy. In *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, Roma, Italia, April 2007. (video session).

[57] N. Mansard, A. Remazeilles, and F. Chaumette. Continuity of varying-feature-set control laws. Technical Report 1864, INRIA/IRISA, September 2007.

[58] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi. Saisie guidée par la vision pendant la marche d'un robot humanoïde. In *2de Journées Nationales de la Robotique Humanoïde, JNRH'07*, Montpellier, France, March 2007.

[59] N. Mansard and F. Chaumette. Enchaînement de tâches en asservissement visuel : application à l'évitement des butées articulaires. In *Congrès jeunes chercheurs en Vision par ordinateur, ORASIS'2005*, Clermont-Ferrand, France, May 2005.

[60] J. Solà, C. Roussillon, N. Mansard, et al. RT-SLAM homepage, 2010. http://www.openrobots.org/wiki/rtslam/.

Thesis

[61] N. Mansard. *Enchaînement de tâches robotiques (in French).* PhD thesis, Univ. Rennes 1, Rennes, France, December 2006.

Supervision

[62] F. Keith. *Optimisation du séquencement de tâches avec lissage du mouvement dans la réalisation de missions autonomes ou collaboratives d'un humanoïde virtuel ou robotique.* PhD thesis, Univ. Montpellier 2, Montpellier, December 2010.

[63] L. Saab. *Generating whole body movements for dynamics anthropomorphic systems under constraints.* PhD thesis, Univ. Toulouse, Toulouse, France, November 2011.

[64] S. Hak. *Reconnaissance de tâches par commande inverse.* PhD thesis, Univ. Toulouse, Toulouse, France, December 2011. (in French).

[65] V. Aurousseau. Master's thesis, INSA Strasbourg, France, 2012.

[66] R. Fleurmond. Master's thesis, Univ. Paul Sabatier, Toulouse, 2012.

[67] M. Reis. Optimisation de calculs en dynamique par metaprograming : application au rnea de featherstone. Master's thesis, Supélec, Rennes, France, July 2011. (in French).

[68] O. Ramos. Master's thesis, Master Erasmus Mundi VIBOT, 2011.

[69] M. Bockelet. Master's thesis, École Normale Supérieure de Cachan, France, 2009.

[70] J-C. Renaud. Command using vision/ strength with the humanoid robot hrp. Master's thesis, IFMA Clermont-Ferrand, France, 2009. (supervised by P. Martinet, LASMEA).

[71] C. Renaudin. Master's thesis, INSA Rennes, France, 2005.

The pdf files can be downloaded from `http://homepages.laas.fr/nmansard`.

Seminotics of Motion

My work is aiming at establishing the bases of a semiotics of motion, in order to facilitate the programing of complex robotics systems. The objective is to build a symbolic model of the action, based on the analysis of the numerical functions that drive the motion (control and planning). The methodology comes from the well-known robotics concepts: motion-planning algorithms, control of redundant systems and task-function approach. The originality of the work is to consider the "task" as the unifying concept both to describe the motion and to control its execution.

The document is organized in two parts. In the first part, the task-function control framework is extended to cover all the possible modalities of the robot. The objective is to absorb from the lowest-possible functional level the maximum of uncertainty factors. It is then not any more necessary to model these factors at the higher functional levels. This sensorimotor layer is then used as a basic "*action vocabulary*" that enables the system to be controlled with a higher-level interface. In the second part, this action vocabulary is used to provide a dedicated robotics programing language, to build motion-planning methods and to describe an observed movement.

The proposed methods are generic and can be applied to a various systems, from robotics (redundant robots) to computer animation (virtual avatars). Nonetheless, the work is more specifically dedicated to humanoid robotics. Without forgetting other possible outlets, humanoid robotics provides a tangible applicative and experimental framework. It also leads toward the natural human motion, as presented in the end of the document.

Keywords: Robotic, redundant systems, anthropomorphic mouvement, sensor-based control, task sequencing, obstacle avoidance

Vers une sémiotique du mouvement

Mes travaux se proposent d'étudier les fondements d'une sémiotique du mouvement pour faciliter la programmation de systèmes robotiques complexes. Il s'agit d'établir des modèles symboliques de l'action sur la base de l'analyse des fonctions numériques qui préludent au mouvement (planification et contrôle). Ce travail s'appuie sur la maîtrise de concepts robotiques bien établis en algorithmique de la planification de mouvements, en commande des systèmes redondants et en modélisation de tâches. L'originalité de l'approche consiste à étendre le concept de tâche comme un cadre unificateur pour décrire le mouvement à effectuer et pour réaliser son exécution.

Le document est organisé en deux axes. Dans un premier temps, on cherche à étendre le cadre de la commande par fonction de tâche, de manière à absorber au plus bas niveau un maximum d'incertitudes qui n'auront ainsi plus à être traitées au niveau décisionnel. C'est de cette couche sensori-motrices que j'entends faire émerger un « vocabulaire de l'action » permettant une commande « haut niveau » du système. Dans un second temps, on s'intéresse à l'organisation de ce vocabulaire pour la programmation et la plannification automatique de mouvement, ou encore comme cadre descripteur de mouvements observés.

Les méthodes proposées se veulent génériques. Elles peuvent être appliquées à une grande variété de systèmes allant de la robotique (robots redondants) à l'animation graphique (avatars virtuels). Néanmoins, le travail est délibérément orienté plus spécifiquement vers la robotique humanoïde. Sans condamner d'autres débouchés, cela permet de fixer à la fois un cadre expérimental et un cadre applicatif précis, et comme présenté dans la fin du document, d'ouvrir le projet vers la thématique plus générale du mouvement humain.

Mots-clefs : Robotique, systèmes redondant, mouvement anthropomorphe, asservissement référencé capteur, enchaînement de tâches, évitement d'obstacles