Contribution à la planification de mouvement pour robots humanoïdes

Oussama Kanoun

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Automatique*

Présentée et soutenue par *Oussama Kanoun*
Le *26 octobre 2009*

Titre : *Contribution à la planification de mouvement pour robots humanoïdes*

JURY
Oussama Khatib Rapporteur
Philippe Bidaud Rapporteur
Jean-Paul Laumond Examinateur
Etienne Ferré Examinateur
Pier-Giorgio Zanone Examinateur
Florent Lamiraux Examinateur

Ecole doctorale : *Systèmes (EDSYS)*
Unité de recherche : *CNRS - Laboratoire d'Analyse et d'Architecture des Systèmes*
Directeur(s) de Thèse : *Jean-Paul Laumond*
Rapporteurs : *Oussama Khatib (Stanford University) et Philippe Bidaud (UPMC)*

To Neila, Foued and Karim

Acknowledgements

This work has been part of the research project Zeuxis for which we gratefully acknowledge a sponsorship from The EADS Foundation.

I owe the fantastic experience I had during my thesis to people of great caliber. First, I realize how big a chance it was to work under the supervision of Dr.Jean-Paul Laumond whom I thank for his guidance and accommodating management. I had the chance to work with Dr.Florent Lamiraux whose scientific advice was a constant reason behind successes. The fact that I could easily concentrate on my thesis topic was largely due to the support of Dr.Anthony Mallet, author of a great robotics software architecture. The teamwork we put forward to achieve vision-guided grasp for HRP-2 was one of the best moments of this thesis. I would like to thank Dr.Eiichi Yoshida who directed me to a lot of essential material and provided friendly support and counsel. I also had the chance to collaborate with Dr.Pierre-Brice Wieber from whom I learnt a lot.

Like the legendary singer said, *I get by with a little help from my friends*. So thank you Anh, Claudia, Wael, Gustavo, Thierry, Brice, Ali, Mathieu, Seb, Carlos, Manish, Pancho, Diego, David, Layale, Thomas and Duong. Special thanks go to Zen-master Minh :). Special thanks to you Antoine and Muriel, for your constant friendship and fantastic meals :)! Grazie mille Soraya, Jérôme, Virginie, Marrrie and Diarmait.

Big thanks fly to my adorable family in Tunis and Sfax and everywhere else in the world.

I owe and dedicate this achievement to my mother Neila, my father Foued and my brother Karim who have always had infinite unconditional support, encouragement and trust. Thank you for your love, far better than anything in this world.

Contents

1 Introduction

1.1 Problem statement

Autonomous robots concretize a *Perception-Planning-Action* loop. *Perception* is the construction of a useful representation of the robot and its environment. This process relies on data produced by cameras, range sensors, force sensors, gyroscopes, etc. *Actions* designate the events that the robot can provoke such as signals (light, voice), motions (involving motors, pneumatics), etc. *Planning* is the process that maps the perception to actions. In this thesis, we will be exclusively focusing on planning problems for humanoid robots.

The planning component of a humanoid robot designed after the human form is a great challenge to roboticists. In front of a robot with a human shape walking and moving dexterously, anyone would rightfully expect a little intelligence in the package. Intelligence, for a machine, is the capacity to perform tasks normally requiring human intelligence (*Oxford dictionary*). All there seems to do to test this intelligence is to submit a task and compare the robot's actions to one's own. This is the principle of The Turing Test. The robot should use all the resources at its disposal and adopt a course of actions that a human would expect from his peers.

Intelligence can be perceived at very simple levels. For instance, to grasp a couple of cups standing close on a table, the human would probably use both hands simultaneously if possible. A humanoid robot with the same possibility taking a cup after the other may probably be dismissed from the closed circle of intelligent robots. Intelligence can be perceived on the level of the action itself: if the table happens to be a little low, instead of decomposing the action into leaning forward on the legs then stretching the arms to grasp the cups, all involved parts of the robot could move simultaneously to achieve the task.

In both examples, intelligence is reflected through the coordination of several resources to save time. In the second example, the robot used its legs to help a manipulation. We might have little conscience of the motion of our lower bodies when performing such manipulation tasks. We may also have little conscience of the infinity of ways in which we could have solved the same problem. One important aspect of the intelligence of humanoid robots must therefore be a seamless exploitation of their rich and redundant mechanical structures.

Unlike humans who might indulge in walking for their pleasure or because it sustains their health, the humanoid robot should be motivated by a precise task for obvious energy-related considerations. How do we relate the steps that we make to the tasks that we perform with our hands or any other part of our bodies? Again, when we think of the number of solutions we realize that the robot must often choose from an infinity, which it still has to be aware of. Let us again consider the simple example of picking up an object, from the floor this time. We have not yet spoken of a level of intelligence to build a piece of furniture from a famous Swedish brand, we will merely ask the robot to collect the fallen assembly manual. The robot, nonetheless, faces various difficulties: where to stand to be able to perform the task? The object is on the ground, therefore it will probably need to crouch, so how to take this posture into account for the motion plan? Is the task feasible at all given its body shape? We are hinting at a second seamless form of intelligence: the humanoid robot must decide if it needs to resort to locomotion in order to carry out a task and if so, find a suitable one.

In fact, most animals possess this level of *intelligence*, they flawlessly move to where an intended action could be fulfilled. The motion of legged forms on a terrain could roughly be seen as a monotonous translation and/or rotation to attain a remote objective. However, a fine adjustment occurs at the end of the locomotion so that the posture is adapted to the intended action. The complexity of actions and postures is much higher for bipeds than for four-legged animals, therefore any heuristic-based resolution of the task-driven locomotion problem is too restrictive.

The planning component of a humanoid robot must rigorously relate its tasks to its whole body and locomotion capabilities, otherwise even the simplest possible tasks could be failed. We aimed in our work at answering this requirement.

1.2 Related works and contributions

The two major components of this topic, whole body motion generation and locomotion planning, have been subject to an extensive research activity. Both subjects could be considered new in the overall young field of robotics. This thesis advances a new view for each subject.

In the context of robot motion control, a *task* is defined as a desired kinematic or dynamic property in the robot (e.g position of hand, forces applied on manipulated object, direction of cameras, etc). For robotic arms with a few degrees of freedom, we may find analytical formulas giving the unique control satisfying a desired task. For highly-articulated structures like humanoid robots, the kinematic structure

is often redundant with respect to a task, offering an infinity of possible controls to choose from. One efficient way of solving the problem of control in redundancy is by identifying a task to the minimum of a convex cost function that can be reached using a numerical algorithm. If we wanted to take advantage of the redundancy of the system and assigned several tasks at a time, we could still resort to the same algorithm and minimize the sum of the cost functions associated with the tasks. However, if the tasks became conflicting, the numerical algorithm could lead to a robot state where none of them are satisfied.

To avoid such situations, it has been proposed to strictly prioritize tasks so that when such a conflict occurs, the algorithm should privilege the task with higher priority. Such a prioritization of a task A over a task B has been modeled in [Liégeois 1977] as the control for task B within the set of controls satisfying task A. The framework was further developed in [Nakamura 1990] and extended to any number of tasks in [Siciliano and Slotine 1991; Baerlocher and Boulic 1998], with illustrations in velocity-based [Yoshida et al. 2006; Mansard and Chaumette 2007; Neo et al. 2007] and torque-based [Khatib et al. 2004] control of humanoid robots.

This classical prioritization algorithm presents an important shortcoming: its original design is unable to consider unilateral constraints, inequalities. And physical systems are always subject to such constraints, due to bounded control values, limited freedom of motion because of obstacles, etc... Sometimes, tasks can also be expressed more rigorously with inequalities. For example, the task "Walk through this corridor" is equivalent to "Go forward while staying in the region delimited by the walls of the corridor". Some workarounds have been proposed, associating for example to each inequality constraint an artificial potential function which generates control forces pushing away from the constraint [Khatib 1986; Marchand and Hager 1998; Chaumette and Marchand 2001], yet inequality constraints would appear then to have the lowest priority. To address this problem, it has been proposed [Mansard et al. 2008] to calculate a weighted sum of controls, each correpsonding to a stack of prioritizd tasks where a subset of inequality constraints is treated as equality constraints. The main problem with the algorithm is its exponential complexity in the number of inequalities. In a third approach, a Quadratic Program (QP) is used to optimize a sum of cost functions, each one associated with a desired task, under strict equality and inequality constraints [Zhao and Badler 1994; Hofmann et al. 2007; Decré et al. 2009; Salini et al. 2009]. However, this approach can consider only two priority levels, the level of tasks that appear as strict constraints of the QP, and the level of tasks that appear in the optimized cost function. We strived to overcome this limitation.

The first outcome of this thesis work is a generalized prioritization algorithm for both equality and inequality tasks for any type of control parameters.

The second part of the work focused on task-driven locomotion planning. In the literature, locomotion planners that are free of ad-hoc strategies are exclusively relying on probabilistic search algorithms. In this approach, the parameters of the problem are found by random exploration of the entire parameter space. A first application for humanoid robots was shown by [Kuffner et al. 2002] to build dynamically stable joint motion with a single foot displacement. [Escande et al. 2009] used the same class of algorithms to plan successive contact ports between the robot and its environment yielding the desired robot task. With the increase of average precessing power, applying search algorithms on high-

dimensional systems such as humanoid robot has become affordable. Nonetheless, we believe that these powerful methods should be saved for complex situations where a local strategy does not suffice.

Some works attempted a different use of random search algorithms[Yoshida et al. 2007; Diankov et al. 2008]. The humanoid is viewed as a wheeled robot that must join a goal position and orientation linked to the task in mind. The search algorithm is parametrized to guarantee collision-free stepping along the solution path. An independent method takes over to plan stable stepping motions along the path. The advantage of this approach is the reduction of the dimension of the problem down to three (two translations and one rotation for each node of parameters in the searched space). It needs, however, a reliable inference of the goal position and orientation from the task and the robot's geometry. In a trial to compensate for this drawback, a two-time strategy has been proposed by[Yoshida et al. 2007]: first infer a gross goal position and orientation for the given task, then plan a path to it and finally determine if there is a need to fine-tune the position and orientation by a single step based on a task-specific strategy. The advantage of this method is to tackle the problem in progressive difficulty. Nonetheless, the series of subproblems suffers from the performance of the weakest link which is the final local strategy used to adjust the stance.

The second outcome of this thesis work is a generic task-driven local footsteps planner that will advantageously replace any task-specific or robot-specific strategy on a flat terrain.

1.3 Outline of the thesis

In chapter 2, the classical prioritized inverse kinematics framework is reviewed for velocity-resolved control of kinematic structures. The algorithms of this framework will be closely related to optimization formulations. Definitions of equality tasks and constraints will also be given.

In chapter 3, we generalize the prioritization framework to any set of equality and inequality tasks and constraints. We will pose the optimization problems and characterize the corresponding solution sets. We end this chapter by giving a stage-by-stage resolution algorithm, as in the classical prioritization framework.

In chapter 4 we illustrate the generalized framework with two scenarios for the humanoid robot HRP-2. The scenarios are based on several tasks and constraints which are also presented along with useful calculus for the numerical resolution. Attention will be especially put on collision avoidance and relaxed center of mass constraints

In chapter 5, a dynamic walk planning method is reviewed. We will present a contribution lying in the merger of two frameworks: numerical inverse kinematics and Zero-Momentum-Point-based control of dynamic biped walk. In the robot experiments that illustrate this work, only dedicated stepping strategies are designed to help tasks as needed.

In chapter 6, we are ready to formulate and solve the problem of interest: where to stand and what footsteps to adopt to carry out the requested tasks. We will use the algorithm presented in chapter 3 and

the tasks and constraints displayed in chapter 4 to formulate an inverse kinematics problem. We will illustrate the efficiency of this methods through a variety of scenarios.

In the final chapter, we exploit whole-body motion generation reviewed in chapters 3 and 4, the footsteps planner seen in chapter 6 and the locomotion planner from chapter 5 to design a local planning component for humanoid robots on flat terrain.

1.4 Associated publications and software

Publications

Kanoun O., Yoshida E. and Laumond J-P. “An optimization formulation for footsteps planning” *IEEE-RAS International Conference on Humanoid Robots, 2009*

Kanoun O., Lamiraux F., and Wieber P.B. “Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots” *IEEE International Conference on Robotics and Automation, 2009*

Yoshida E., Kanoun, O., Esteves Jaramillo C., and Laumond J.P. “Task-driven support polygon reshaping for humanoids” *IEEE-RAS International Conference on Humanoid Robots, 2006*

Yoshida E., Mallet A., Lamiraux F., Kanoun O., Stasse O., Poirier M., Dominey P.F., Laumond J.P. and Yokoi K “”Give me the purple ball” - he said to HRP-2 N.14” *IEEE-RAS International Conference on Humanoid Robots 2007*

Kanehiro F., Lamiraux F., Kanoun O., Yoshida E. and Laumond J.P. “A local collision avoidance method for non-strictly convex objects” *Robotics: Science and Systems Conference 2008*

Yoshida E., Poirier M., Laumond J.P., Kanoun O., Lamiraux F., Alami R. and Yokoi K.“Whole-body motion planning for pivoting based manipulation by humanoids” *IEEE International Conference on Robotics and Automation 2008*

Kanehiro F., Yoshida E., Lamiraux F., Kanoun O. and Laumond, J.P. “A local collision avoidance method for non-strictly convex object” (in japanese) *Journal of the Robotics Society of Japan 2008*

Yoshida E., Laumond J.P., Esteves Jaramillo C., Kanoun O., Sakaguchi T., and Yokoi K. “Whole-body locomotion, manipulation and reaching for humanoids” *Lecture Notes in Computer Science 5277, Springer, 2008, ISBN 978-3-540-89219-9*

Yoshida E., Poirier M., Laumond J.P., Kanoun O., Lamiraux F., Alami R. and Yokoi K. “Regrasp planning for pivoting manipulation by a humanoid robot” *IEEE International Conference on Robotics and Automation 2009*

Created software

hppGik : is a software implementing primarily a prioritized equality system solver (Problem 2.19, Algorithm 2). A second solver dedicated to prioritized inverse kinematics is built on top (Algorithm 3). Joint limits are taken into account following (2.22). Singularities are with using either the regulated pseudo-inversion (2.10) or the thresholded one (2.9). A set of equality tasks (body position, orientation, parallelism, plane, posture, gaze direction and center of mass position) is also implemented. The dynamic walk generation based on ZMP trajectory planning (section 5.1.4) is also integrated. There are objects that enable the user to program desired tasks on a time line, including dynamic locomotion tasks expressed as step tokens, and launch the batch resolution of the entire time line.

hppHik : is a software that implements the novel linear equality and inequality prioritization (Algorithm 4). It is based on the a quadratic program solver courtesy of AEM-Design[Lawrence et al.]. It implements the slack variable-based optimization of linear inequality systems (3.8).

hppLocalStepper : implements the footsteps planner (currently Block 1 of Algorithm 7 only). All tasks presented in chapter 4 are implemented, including collision avoidance and static equilibrium with a deformable robot support polygon.

2

Classical prioritized inverse kinematics

2.1 Definitions

Kinematics is a branch of mechanics that studies the motion of solid particles without consideration of their inertia and the forces acting on them. To introduce inverse kinematics let us consider the kinematic chain composed of three links represented in figure 2.1.

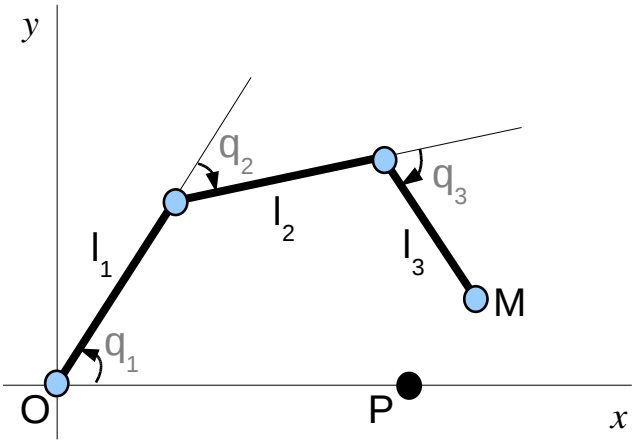


Figure 2.1: A three-link kinematic chain

Three rotation joints make the chain move in the plane $(O,\vec{x},\vec{y})$, where the position of the point M is :

$$\overrightarrow{OM} = \begin{pmatrix} x_M \\ y_M \end{pmatrix} = \begin{pmatrix} l_1 cos(q_1) + l_2 cos(q_1+q_2) + l_3 cos(q_1+q_2+q_3) \\ l_1 sin(q_1) + l_2 sin(q_1+q_2) + l_3 sin(q_1+q_2+q_3) \end{pmatrix} \tag{2.1}$$

Calculating the position of M, tip of the third link, from the *joint configuration* (q_1, q_2, q_3) is a *forward kinematics problem*. It consists in evaluating a property of the chain in *the work space* given the configuration of the chain in *the joint space*, whose dimension equals the number of degrees of freedom of the kinematic structure. *Inverse kinematics* designates the opposite problem consisting in finding a configuration fulfilling certain properties, for instance positioning the tip M at the point P.

Considering any kinematic structure with n degrees of freedom, we define its configuration:

$$q = (q_1, \cdots, q_n) \in \Re^n$$

and express desired values for some kinematic properties, which can be written without loss of generality as:

$$x(q) = \vec{0} \tag{2.2}$$

where:

$$\begin{array}{rccc} x: & \Re^n & \longrightarrow & \Re^m \\ & q & \longmapsto & x(q) \end{array}$$

is the m-dimensioned vectorial function defining a kinematic property. For a configuration q of the structure, if (2.2) is verified and is to be kept, it can be interpreted as a kinematic **equality constraint**. In the opposite case where the relationship is not yet fulfilled, we may call it a kinematic **equality task**. A task expressed as (2.2) is often a non linear function of q that does not admit a trivial inverse. In complex kinematic structures such as humanoid robots, the degrees of freedom relevant to a given task often exceed in number the dimension of the task. For such systems we resort to numerical methods to solve inverse kinematics problems.

2.2 Differential kinematics

By *differential kinematics*[Nakamura 1990] we refer to the iterative process of computing small configuration updates for kinematic structure to converge towards a state that achieves a given task. This process is also known as the Newton-Raphson algorithm applied to inverse kinematics. The tasks we consider here are differentiable vector functions of the joint configuration.

By computing the jacobian of the task $J = \frac{\partial x}{\partial q}(q)$, we can calculate configuration updates δq to make the task value converge towards $x(q) = 0$. These joint updates are solution of the following linear differential equation [Liégeois 1977]:

$$J\delta q = -\lambda x(q) \tag{2.3}$$

where λ is a positive real number. To simplify the notations, we define:

$$\delta x\,(q, \lambda) = -\lambda x(q)$$

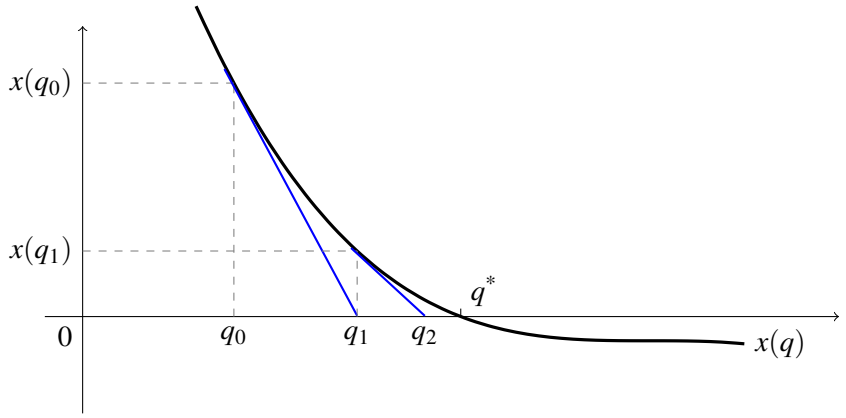


Figure 2.2: Newton-Raphson iterations to solve $x(q) = 0$

and rewrite the previous equation simply as:

$$J\delta q = \delta x \tag{2.4}$$

The relationship (2.4) is a system of linear equations for which a solution might not always exist, depending on the rank of J. If J keeps a full rank, successive updates make the configuration converge to q^* satisfying $x(q^*) = 0$. We illustrate this process (Algorithm 1) in figure 2.2 using a simple function mapping a 1D configuration space to a 1D workspace.

Algorithm 1 Numerical inverse kinematics for a single task $T(q) = 0$

1: Define $\varepsilon_p > 0$ the minimum task value progression required to continue
2: Initialize $p > \varepsilon_p$
3: **while** $p > \varepsilon_p$ **do**
4: Evaluate $T_0 \leftarrow \|T(q)\|$
5: Calculate jacobian of $T(q)$: J_T
6: Solve $J_T \delta q = -\lambda T(q)$ in δq
7: Update configuration $q \leftarrow q + \delta q$
8: Evaluate $T_1 \leftarrow \|T(q)\|$
9: Calculate task progression: $p = T_0 - T_1$
10: **end while**

We suppose that J has full rank (the opposite case is studied in next section). The solution of the linear system (2.4) form an affine subspace of $\Re^n$. For instance, in case $n = 3$ this affine subspace is either a point, a line, a plane or the whole space $\Re^3$. One way of calculating a point in that subspace is to orthogonally project the origin of the updates space on the affine subspace. This projection is the result of the following minimization:

$$\begin{cases} \min\limits_{\delta q} & |\delta q\|^2 \\ s.t & J\delta q = \delta x \end{cases} \tag{2.5}$$

which yields:

$$\begin{aligned} \delta q &= J^{\#}\delta x \\ &= J^T(JJ^T)^{-1}\delta x \end{aligned} \tag{2.6}$$

$J^{\#}$ is called the pseudo-inverse of J. A solution to the linear system (2.4) in general can be written as:

$$\delta q = J^{\#}\delta x + (I - J^{\#}J)z \tag{2.7}$$

where z is any vector in $\Re^n$. The operator $(I - J^{\#}J)$ is the projector along the direction of the affine subspace of solutions. Figure 2.2 illustrates the components of the general form.

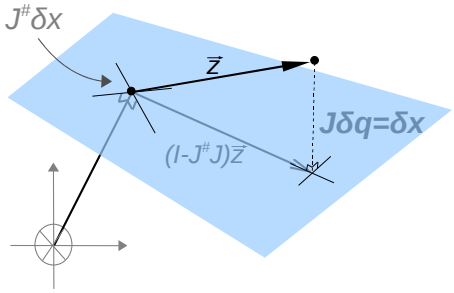


Figure 2.3: The plane represents the solutions to a linear equality in a 3D space

When the linear system (2.4) is over-constrained, the minimization of $\|J\delta q - \delta x\|^2$ is considered instead, yielding the solution:

$$\delta q = (J^T J)^{-1} J^T \delta x \tag{2.8}$$

2.3 Local extrema, or singularities

Gradient descent methods, such as Newton-Raphson algorithm, are subject to local extrema. Consider for example the case represented in figure 2.4. The function x has a local minimum whose neighborhood does not intersect the subspace $x(q) = 0$. At the minimum, the jacobian becomes singular, the product JJ^T gives an ill-conditioned matrix, which results in the divergence of the numerical method (see how q_2 falls far from the local optimum).

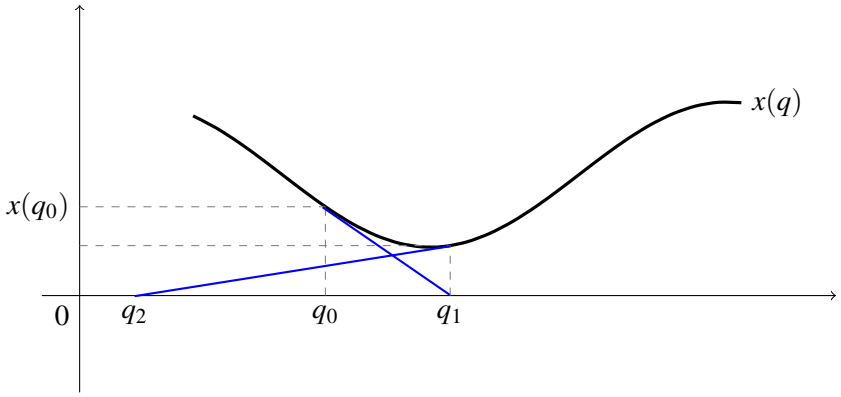


Figure 2.4: Divergence of Newton-Raphson algorithm near singularity

Near singularity, one method[Nakamura 1990] to avoid divergence is to perform a singular value decomposition on J and only invert the well conditioned part, discarding the singular directions. To do so, we decompose J as:

$$J = U \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \end{pmatrix} V^T$$

then cancel the singular values below a chosen threshold and invert the ones above

$$J^{\#} = V \begin{pmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T \tag{2.9}$$

A second method[Nakamura and Hanafusa 1986] consists in replacing the optimization problem (2.5) by the following:

$$\begin{cases} \min\limits_{w,\delta q} & \|w\|^2 + k^2 \|\delta q\|^2 \\ s.t & J\delta q - \delta x = w \end{cases} \tag{2.10}$$

where k is a scalar. Minimizing the first part of the objective function tends to satisfy the linear constraint while minimizing the second part tends to keep a null update. Unless the linear constraint admits the null vector as solution, in which case it represents the optimum, the outcome of the optimization is a joint update with smaller norm than the one from (2.6). The importance of the second part of the objective function is scaled with k, directly influencing the norm of the optimal point, which is given by the formula:

$$\delta q = J^T (JJ^T + k^2 I)^{-1} \delta x \tag{2.11}$$

The factor k^2 amplifies all eigen values of JJ^T thus *regulating* the pseudo-inversion. To picture this process, consider a real function x and the tangent to its curve at point (x_0, y_0):

$$x'(q - q_0) = x - x_0$$

When the derivative of x is not null, damping the inverse of x by a factor k^2 changes the tangent to:

$$\frac{(x')^2+k^2}{x'}(q-q_0)=x-x_0$$

We illustrate in figure 2.5 the effect of factor k^2 on the tangents near the singularity.

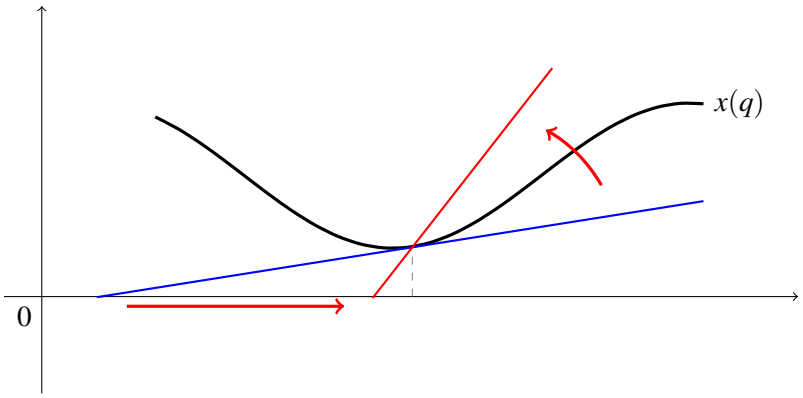


Figure 2.5: Regulated pseudo-inversion: steeper tangents prevent large joint updates near singularity.

2.4 Prioritization

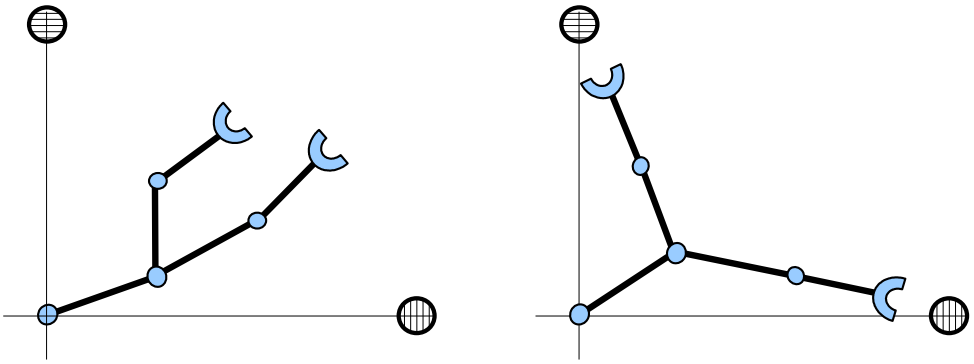


Figure 2.6: One target has to be prioritized to prevent failing both grasps.

Suppose that we seek the completion of two tasks $x_1(q)=0$ and $x_2(q)=0$ and that we cannot assess their feasibility without resorting to numerical resolution. Suppose, furthermore, that task 1 is more important that task 2. We derive the linear equations for differential kinematics:

$$J_1\delta q_1=\delta x_1 \tag{2.12}$$

$$J_2\delta q_2=\delta x_2 \tag{2.13}$$

Equation (2.12) has an affine subspace of solutions:

$$\left\{\delta q \in \Re^n \; s.t \; \delta q = {J_1}^{\#}\delta x_1 + (I - {J_1}^{\#}J_1)z, \; z \in \Re^n\right\} \tag{2.14}$$

To express the absolute priority of task 1 with respect to task 2, δq is replaced in (2.13) giving the linear system:

$$\delta x_2 - J_2 {J_1}^{\#}\delta x_1 = J_2(I - {J_1}^{\#}J_1)z \tag{2.15}$$

Defining $N_1 = (I - {J_1}^{\#}J_1)$ and $\delta q_1 = {J_1}^{\#}\delta x_1$, the above linear system is solved in z:

$$\begin{cases} \min\limits_{z} & \|\delta z\|^2 \\ s.t & \delta x_2 - J_2\delta q_1 = J_2 N_1 z \end{cases} \tag{2.16}$$

yielding:

$$\delta q = {J_1}^{\#}\delta x_1 + N_1(J_2N_1)^{\#}(x_2 - J_2\delta q_1)$$

which can be simplified to ([Nakamura 1990]):

$$\delta q = \overbrace{\underbrace{{J_1}^{\#}\delta x_1}_{\delta q_1} + (J_2N_1)^{\#}(x_2 - J_2\delta q_1)}^{\delta q_2} \tag{2.17}$$

Figure 2.4 shows a representation of the successive orthogonal projections that lead to δq. To express the priorities, the second linear system is solved within the solutions of the first linear system, hence the second projection occurring inside the plane defined by the solutions of the first system. In case the tasks

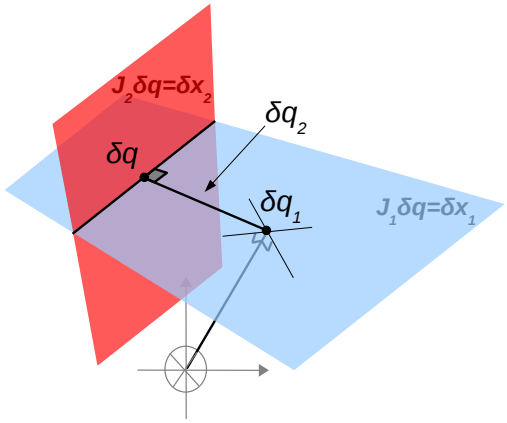
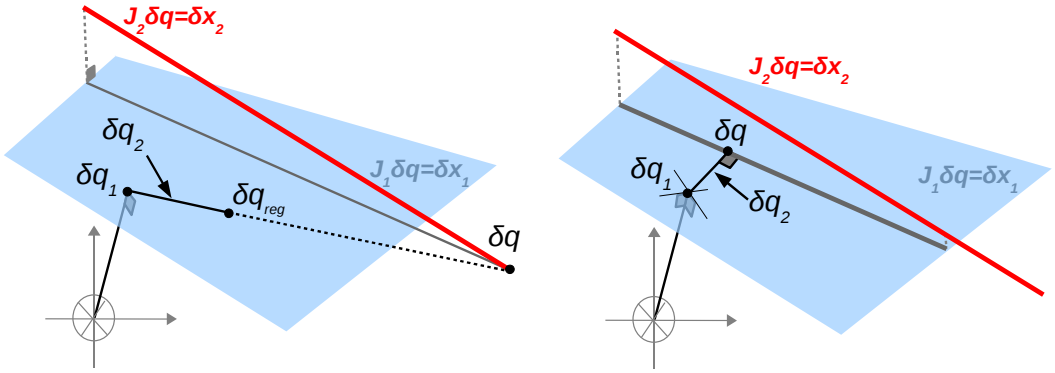


Figure 2.7: Compatible tasks. The linear equality systems are compatible thus a solution satisfying both is found by successive pseudo-inversions.

are conflicting the union of linear systems is singular. The joint update δq satisfies the first linear system and cannot satisfy the second. The regulated (2.11) or truncated (2.9) pseudo-inversion of the second linear system ensures that the singularity does not yield a large update. The effect of both methods is



(a) Regulation of the second pseudo-inversion gives a point between the solution to the first linear system and the solution to the normal pseudo-inversion of the second linear system.

(b) Truncation of singular directions. The projection of the red line onto the plane represents the set of solutions to the truncated system.

Figure 2.8: Conflicting tasks. The red line represents the solutions of the linear system corresponding to the second task. The singularity of second task is represented by the near parallelism of the red line and the plane.

shown in figure 2.4.

The prioritization process is straightforwardly generalized to a set of k tasks. At priority level i, the solved optimization problem is:

$$\begin{cases} \min\limits_{w,\delta q_i} & \|w\|^2 + {k_i}^2 \|\delta q_i\|^2 \\ s.t & J_i N_{i-1} \delta q_i - (\delta x_i - J_i \delta q_{i-1}) = w \end{cases} \tag{2.18}$$

Based on this generalization, Algorithm 2 describes the steps taken to solve a set of prioritized linear equality systems. We propose the following compact formulation of the problem solved by this algorithm:

Find δq^* solution to the optimization problem:

$$\begin{cases} \min\limits_{\delta q \in S_k} & \|\delta q\|^2 \\ \text{s.t} & S_0 = \Re^n \\ & S_i = \underset{\delta q \in S_{i-1}}{\text{Arg min}} \ \|J_i \delta q - \delta x_i\|^2 \end{cases} \tag{2.19}$$

Algorithm 2 calculates an update that acts on the values of all tasks simultaneously. Between two consecutive stages ranked i and $i+1$, the joint update is modified to take into account the tasks at priority level i without compromising the the tasks from level 1 to $i-1$. Like for the single task case, the linear systems derived from the prioritized tasks are repeatedly solved by Algorithm 2 until convergence. The convergence criterion takes into account all the tasks and observes their priority. It is shown in Algorithm 3.

Algorithm 2 Solve k prioritized linear equality systems

n is the dimension of the kinematic structure
$N_0 \leftarrow I$ (n by n identity matrix)
$\delta q \leftarrow 0$ (n-sized null vector)
for $i = 1$ to k **do**
    Calculate linear equality system (J_i, δx_i)
    $\hat{J}_i \leftarrow J_i N_{i-1}$
    Calculate pseudo-inverse $\hat{J}_i^{\#}$
    $\delta q_i \leftarrow \hat{J}_i^{\#}(\delta x_i - J_i \delta q_{i-1})$
    $\delta q \leftarrow \delta q + \delta q_i$
    $N_i \leftarrow N_i - \hat{J}_i^{\#} \hat{J}_i$
end for

2.5 Handling inequality constraints

Numerical inverse kinematics are used to control the motion of virtual characters, humanoid robots, etc. Most actuators on robotic systems only allow for a bounded range of motion between the connected bodies. Virtual characters also require joint limits to be observed for realism. Joint limits are naturally expressed as inequality constraints on the configuration:

$$inf(q) \leq q \leq sup(q) \tag{2.20}$$

and translate to joint update space as:

$$inf(q) - q \leq \delta q \leq sup(q) - q \tag{2.21}$$

which define an admissible convex volume in joint space outside of which a point corresponds to an update that would violate the joint limits. Naturally, the desired bounds for δq may be more constraining than in (2.21). From a geometrical point of view, pseudo-inversion is the computation of the orthogonal projection of a point onto an affine subspace. Nothing prevents the coordinates of the projection to fall outside the convex volume allowed by inequalities (2.21). Nonetheless, pseudo-inversion of a linear equality system is a fast operation that can be afforded for online robot control, animation of virtual figures in movies and video games. Therefore, modified versions of Algorithm 2 were proposed to account for joint limits.

One solution is to replace the optimization problem at every priority stage by the following one:

$$\begin{cases} \min\limits_{\delta q} & \delta q^T W \delta q \\ s.t & J\delta q = \delta x \end{cases} \tag{2.22}$$

Algorithm 3 Numerical inverse kinematics for k prioritized tasks $\{T_1(q) = 0, ..., T_k(q) = 0\}$

Define ε_p the minimum task progression required to continue
Call p the measure of task value progression
repeat

  for i from 1 to k **do**
    Calculate value of task i: $T_i^0(q)$
  end for

  Solve the k prioritized linear systems [Algorithm 2]
  Update configuration $q \leftarrow q + \delta q$

  for i from 1 to k **do**
    Calculate value of task i: $T_i^1(q)$
    Evaluate progression $p = T_i^1(q) - T_i^0(q)$
    if $p > \varepsilon_p$ **then**
      break.
    end if
  end for

until $p < \varepsilon_p$

W is a positive diagonal n-by-n matrix. The objective function here affects each degree of freedom in the configuration q with a weight w_i. To emphasize this, the objective function can be written as:

$$\sum_{i=1}^{n} w^i {\delta q^i}^2$$

where δq^i and w^i are the i-th component of δq and w respectively. The larger the weight w_i, the more $|\delta q^i|$ is minimized with respect to other coordinates, hence the idea: if $\frac{dq^i}{dt} > 0$ and q^i is close enough to $sup(q)$ then w^i can be gradually increased to reduce the norm of δq^i, making the corresponding joint brake before violating the upper limit. The same reasoning holds for the lower limit. The analytical solution to problem (2.22) is:

$$\begin{aligned} \delta q &= J_W^{\#} \delta x \\ &= W^{-1} J^T (J W^{-1} J^T)^{-1} \delta x \end{aligned} \tag{2.23}$$

The weighted pseudo-inverse $J_W^{\#}$ is what we would have obtained from the simple pseudo-inversion of the system:

$$J\sqrt{W^{-1}}\delta q = \delta x \tag{2.24}$$

Expanding above equation into:

$$\sum_{i=1}^{n} \frac{1}{\sqrt{w^i}} \frac{\partial x}{\partial q^i} \delta q^i = \delta x \tag{2.25}$$

we could observe that the factor $\frac{1}{\sqrt{w^i}}$ scales the partial derivative of the task x, acting as *brakes* or *accelerator* on the coordinate q^i. The performance of this method relies essentially on the tuning of

the parameters w^i with respect to the state of the kinematic structure.

The above method does not address the general case where any inequality constraint on δq might be considered. For example, bounding the velocity of a point P in the work space along vector d to remain below a maximal value c is written as:

$$d^T \delta P(q) \leq c$$

from which an inequality constraint on the joint update follows:

$$d^T J_P \delta q \leq c$$

where J_P is the jacobian of the position of point P with respect to q.

There are solutions proposed by [Baerlocher 2001; Peinado et al. 2005] to account for linear inequality constraints within Algorithm 2. The inequality constraints are monitored after each priority stage and if violated, the stage is re-computed while taking the violated inequality constraints as hard equality constraints. This is the principle of active set algorithm. Supposing that priority stage i sees the system of inequality constraints $A\delta q \leq b$ violated, the following optimization problem is solved instead of the usual problem (2.18):

$$\begin{cases} \min\limits_{w,\delta q_i} & \|w\|^2 + {k_i}^2 \|\delta q_i\|^2 \\ s.t & J_i N_{i-1} \delta q_i - (\delta x_i - J_i \delta q_{i-1}) = w \\ & A\delta q = b \end{cases} \tag{2.26}$$

The resolution of 2.26 is repeated until the solution update of stage i, δq_i, satisfies all the inequality constraints. The algorithm resumes its normal course afterward.

The main drawback of the method is the nested loop of pseudo-inversions that must be done at every stage to enforce the inequality constraints . There are more efficient algorithms to solve linearly constrained optimization problems with quadratic costs. As a matter of fact, other works[Zhao and Badler 1994; Faverjon and Tournassoud 1987] resorted to these algorithms. However, the separation of tasks into discrete priority levels was not available and the authors resorted to weighted sums of objective functions corresponding to several simultaneous tasks, such as:

$$\begin{cases} \min\limits_{\delta q} & \sum_{i=1}^{k} w^i \|J_i \delta q - \delta x_i\|^2 \\ s.t & A\delta q = b \\ & B\delta q \leq c \end{cases} \tag{2.27}$$

The weighted objective function does not permit strict enforcement of priority. When two of the linear systems represented in the objective function cannot be solved simultaneously, the solution to the minimization problem is a trade-off update that satisfies neither of the systems.

3

Generalizing priority to inequality tasks

3.1 Inequality task definition

Inequality tasks differ from inequality constraints in the way equality tasks differ from equality constraints. For a kinematic structure with n degrees of freedom, we have a differentiable kinematic property $g: \ q \longmapsto g(q)$ whose values are desired below a certain threshold, expressed without loss of generality in the following inequality:

$$g(q) \leq 0 \tag{3.1}$$

For a configuration q of the structure, if (3.1) is verified and is to be kept, it can be interpreted as a kinematic **inequality constraint**. In the opposite case where the relationship is not yet fulfilled, we may call it a kinematic **inequality task**.

One trivial example of inequality task can be $g(q) = Hz(q) - 0.5$ where $Hz(q)$ is the height of the robot hands from the ground. In this example, the task $Hz(q) - 0.5 \leq 0$ requires the hands to be below the horizontal plane of equation $z = 0.5$. Inequality tasks may be solved with Newton-Raphson algorithm like equality tasks through an iterative resolution of a linear differential system of inequalities. Defining $J_g = \frac{\partial g}{\partial q}(q)$, we write the system to solve at a given configuration q:

$$J_g \delta q \leq -\lambda g(q) \quad \lambda \in \Re^{+*} \tag{3.2}$$

The method of resolution of such systems is detailed in the next section. Continuing with the above example on the hand, equation (3.2) gives the following inequality:

$$\frac{\partial Hz}{\partial q}\delta q \leq -\lambda(Hz(q) - 0.5)$$

For a constant value of λ and when the height of hands is below 0.5, the above equation sets a decreasing upper bound on the vertical hand velocity, a bound that is equal to 0 for $z = 0.5$. Above the plane, the upper bound is negative and the plane $z = 0.5$ acts as an attractor.

A linear inequality, when it has solutions, defines a halfspace. The set of solutions of a system of linear inequalities is the intersection of the half spaces generated by its inequalities. This set is a volume of $\Re^n$ bounded by a convex polytope which may be closed or infinite (for example a half space is infinite). See figure 3.1 for an illustration of a system of linear inequalities in $\Re^2$.

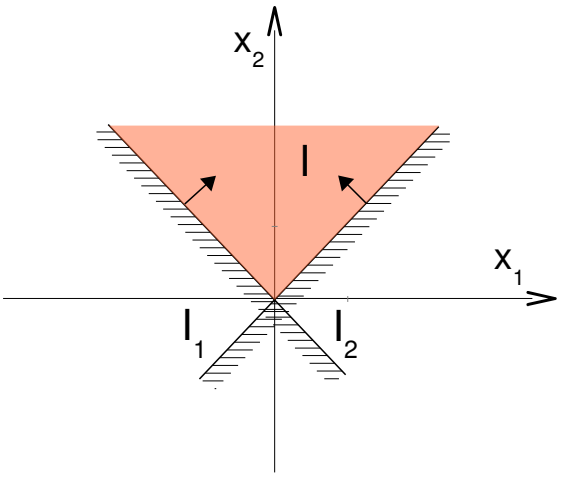


Figure 3.1: The linear inequalities $y \geq x$ and $y \geq -x$ determine the filled convex polytope.

One might observe that any equality task $f(q) = 0$ can be expressed as two simultaneous inequality tasks, $f(q) \geq 0$ and $f(q) \leq 0$. Therefore inverse kinematics problems expressed as inequalities engulf those that use equalities. One attractive aspect of inequalities is the possibility to specify a range or a bounding on a kinematic property. What remains to be constructed is a framework that authorizes the strict prioritization of inequality and equality tasks in any order. One may wonder about the scenarios that require the inequality tasks to be at lower priority than equalities. Consider for example a humanoid robot which has to grasp an object seen with embedded cameras. It is best if its reaching hand does not come between the cameras and the object too soon. This is because we would like to keep checking the visual target to maximize the chance of a successful grasp. In this scenario, the robot has to accomplish a primary reaching task and a secondary region-avoidance task.

3.2 Description of the approach

Affecting priorities to linear systems means that we leave some of the systems unsolved to respect the ones with higher priorities. Letting L_1 and L_2 be two linear systems without common solutions, prioritizing L_1 over L_2 means that we retain a solution which satisfies L_1 to the expense of L_2. Nonetheless, to take into account L_2, one may select a solution of L_1 which minimizes the euclidean distance to L_2's solutions set. Euclidean distance is one example of optimality criterion adapted to systems of linear equalities. As a matter of fact, a point realizing this shortest distance belongs to the orthogonal projection of L_2's solutions on L_1's and can be obtained analytically. Furthermore, the entire set of points realizing the shortest distance may also be determined analytically [Nakamura 1990; Siciliano and Slotine 1991]. To solve a third system of linear equalities L_3, the resolution is done within L_2's optimal set.

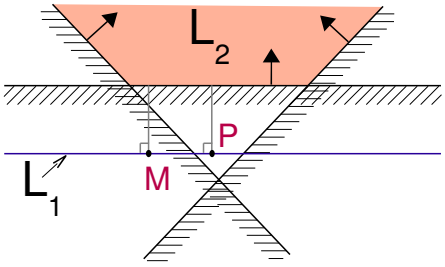


Figure 3.2: The primary linear equality L_1 and a secondary system of 3 linear inequalities L_2 are without common solutions. M and P are solutions of L_1 minimizing the euclidean distance to L_2's set, however, P should be preferred since it satisfies two inequalities out of three while M satisfies none.

For our problem, we adopt the same approach consisting in solving every linear system in the optimal set defined by higher priorities. When we introduce systems of linear inequalities, however, we introduce solution sets which are volumes of $\Re^n$ bounded by convex polytopes. In this particular case, euclidean distance is not a good optimality criterion (Figure 3.2). Therefore, we start by mapping equality and inequality tasks to suitable optimality criteria and study the nature of generated sets of solutions. We prove that the optimizations result in sets that can be described with linear systems and we deduce a resolution algorithm that is relatively easy to implement. This algorithm intended to prioritize inverse kinematics tasks can be straightforwardly applied to any problem involving the prioritization of a set of linear equality and inequality systems, regardless of the type of parameters. For this chapter, we will abandon the notations J and δq to emphasize the generality of the algorithm.

3.3 Mapping linear systems to optimization problems

In this section we construct optimization problems to solve each type of linear system. For each problem, we demonstrate the nature of the optimal set.

Let A and C be matrices in $\Re^{m\times n}$ and b and d vectors in $\Re^m$ with $(m,n) \in \mathbb{N}^2$. We will consider in the following either a system of linear equalities

$$Ax = b \tag{3.3}$$

or a system of linear inequalities

$$Cx \leq d \tag{3.4}$$

or both. When $m = 1$, (3.3) is reduced to one linear equation and (3.4) to one linear inequality.

3.3.1 System of linear equalities

When trying to satisfy a system (3.3) of linear equalities while constrained to a non-empty convex set $\Omega \subset \Re^n$, we will consider the set S_e of optimal solutions to the following minimization problem:

$$\min_{x\in\Omega} \frac{1}{2}\|w\|^2 \tag{3.5}$$

with

$$w = Ax - b. \tag{3.6}$$

Since the minimized function is coercive, the set S_e is non-empty. We also have the property:

$$x_1, x_2 \in S_e \Leftrightarrow x_1, x_2 \in \Omega \text{ and } Ax_1 = Ax_2, \tag{3.7}$$

from which we can conclude that the set S_e is convex.

Proof: Let us consider an optimal solution x^* to the minimization problem (3.5)-(3.6). The gradient of the minimized function at this point is

$$A^T(Ax^* - b).$$

The Karush-Kuhn-Tucker optimality conditions give us that the scalar product between this gradient and any vector v not pointing outside Ω from x^* is non-negative,

$$w^{*T}Av \geq 0$$

with

$$w^* = Ax^* - b.$$

Let us consider now two such optimal solutions, x_1^* and x_2^*. Since the set Ω is convex, the direction $x_2^* - x_1^*$ points towards its inside from x_1^*, so we have

$$w_1^{*T}A(x_2^* - x_1^*) \geq 0$$

which is equivalent to

$$w_1^{*T}w_2^* - \|w_1^*\|^2 \geq 0.$$

The same can be written from x_2^*,

$$w_2^{*T} w_1^* - \|w_2^*\|^2 \geq 0,$$

so that we obtain

$$\|w_2^* - w_1^*\|^2 = \|w_2^*\|^2 + \|w_1^*\|^2 - 2w_2^{*T} w_1^* \leq 0,$$

but this squared norm cannot be negative, so it must be zero and $w_2^* = w_1^*$, what concludes the proof. ■

In the unconstrained case, when $\Omega = \Re^n$, the solutions of (3.5)-(3.6) are such that $A^T Ax^* = A^T b$. This minimization problem corresponds therefore to a constrained pseudo-inverse solution of the system of linear equalities (3.3).

3.3.2 System of linear inequalities

When trying to satisfy a system (3.4) of linear inequalities while constrained to a non-empty convex set $\Omega \subset \Re^n$, we will consider the set S_i of optimal solutions to the following minimization problem:

$$\min_{x \in \Omega, w \in \Re^m} \frac{1}{2}\|w\|^2 \tag{3.8}$$

with

$$w \geq Cx - d, \tag{3.9}$$

where w plays now the role of a vector in $\Re^m$ of *slack variables*. Once again, since the minimized function is coercive, the set S_i is non-empty. Considering each inequality $c^j x \leq b^j$ of the system (3.4) separately, we also have the property:

$$\begin{aligned} x_1, x_2 \in S_i \Leftrightarrow x_1, x_2 \in \Omega \text{ and} \\ \forall j \begin{cases} c^j x_1 \leq d^j \Leftrightarrow c^j x_2 \leq d^j, \\ c^j x_1 > d^j \Rightarrow c^j x_1 = c^j x_2, \end{cases} \end{aligned} \tag{3.10}$$

which means that all the optimal solutions satisfy a same set of inequalities and violate the others by a same amount, and from which we can conclude that the set S_i is convex.

Proof: Let us consider an optimal solution x^*, w^* to the minimization problem (3.8)-(3.9). The Karush-Kuhn-Tucker optimality conditions give that for every vector v not pointing outside Ω from x^*,

$$w^{*T} Cv \geq 0$$

and

$$w^* = \max\{0, Cx^* - d\}. \tag{3.11}$$

This last condition indicates that if an inequality in the system (3.4) is satisfied, the corresponding element of w^* is zero, and when an inequality is violated, the corresponding element of w^* is equal to the value of the violation.

Let us consider now two such optimal solutions, x_1^*, w_1^* and x_2^*, w_2^*. Since the set Ω is convex, the

direction $x_2^* - x_1^*$ points towards its inside from x_1^*, so we have

$$w_1^{*T} C(x_2^* - x_1^*) \geq 0$$

which is equivalent to

$$w_1^{*T}(Cx_2^* - d) - w_1^{*T}(Cx_1^* - d) \geq 0.$$

The optimality condition (3.11) gives

$$w_1^{*T} w_2^* \geq w_1^{*T}(Cx_2^* - d)$$

and

$$w_1^{*T} w_1^* = w_1^{*T}(Cx_1^* - d),$$

so we obtain

$$w_1^{*T} w_2^* - \|w_1^*\|^2 \geq 0.$$

The same can be written from x_2^*,

$$w_2^{*T} w_1^* - \|w_2^*\|^2 \geq 0,$$

so that we obtain

$$\|w_2^* - w_1^*\|^2 = \|w_2^*\|^2 + \|w_1^*\|^2 - 2w_2^{*T} w_1^* \leq 0,$$

but this squared norm cannot be negative, so it must be zero and $w_2^* = w_1^*$, what concludes the proof. ■

3.3.3 Mixed system of linear equalities and inequalities

We can observe that systems of linear equalities and systems of linear inequalities are dealt with optimization problems (3.5)-(3.6) and (3.8)-(3.9) which have similar lay-outs and similar properties (3.7) and (3.10). The generalization of these results to mixed systems of linear equalities and inequalities is therefore trivial and we will consider in the following the minimization problem (in a more compact form)

$$\min_{x \in \Omega, w \in \Re^m} \frac{1}{2}\|Ax - b\|^2 + \frac{1}{2}\|w\|^2 \tag{3.12}$$

with

$$Cx - w \leq d. \tag{3.13}$$

The set of solutions to this minimization problem shares both properties (3.7) and (3.10).

3.4 Prioritizing linear systems

3.4.1 Formulation

Let us consider now the problem of trying to satisfy a set of systems of linear equalities and inequalities with a strict order of priority between these systems. At each level of priority $k \in \{1, \ldots p\}$, both a system of linear equalities (3.3) and a system of linear inequalities (3.4) are considered, with matrices

and vectors A_k, b_k, C_k, d_k indexed by their priority level k. At each level of priority, we try to satisfy these systems while strictly enforcing the solutions found for the levels of higher priority. We propose to do so by solving at each level of priority a minimization problem such as (3.12)-(3.13). With levels of priority decreasing with k, that gives:

$$S_0 = \Re^n, \tag{3.14}$$

$$S_{k+1} = \underset{x\in S_k, w\in\Re^m}{\text{Arg min}} \ \frac{1}{2}\|A_k x - b_k\|^2 + \frac{1}{2}\|w\|^2 \tag{3.15}$$

$$\text{with } C_k x - w \leq d_k. \tag{3.16}$$

3.4.2 Properties

A first direct implication of properties (3.7) and (3.10) is that throughout the process (3.14)-(3.16),

$$S_{k+1} \subseteq S_k.$$

This means that the set of solutions found at a level of priority k is always strictly enforced at lower levels of priority, what is the main objective of all this prioritization scheme.

A second direct implication of these properties (3.7) and (3.10) is that if S_k is a non-empty convex polytope, S_{k+1} is also a non-empty convex polytope, the shape of which is given in properties (3.7) and (3.10). Figures 3.3 and 3.4.2 illustrate how these sets evolve in different cases. Classically, these convex polytopes can always be represented by systems of linear equalities and inequalities:

$$\forall k,\ \exists \bar{A}_k, \bar{b}_k, \bar{C}_k, \bar{d}_k \text{ such that } x \in S_k \Leftrightarrow \left\{ \begin{array}{l} \bar{A}_k x = \bar{b}_k \\ \bar{C}_k x \leq \bar{d}_k \end{array} \right.$$

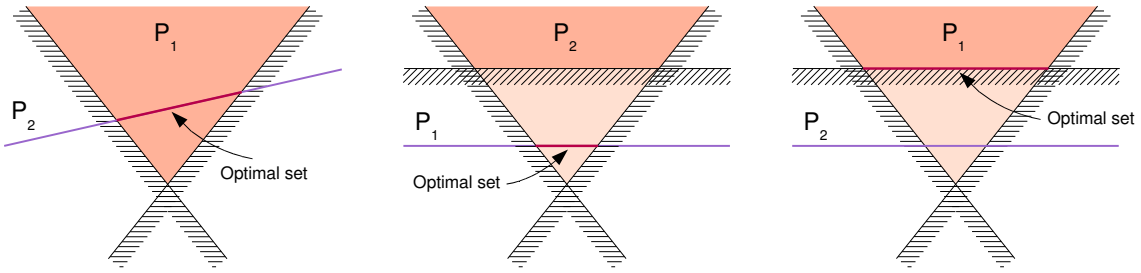
With this representation, the step (3.15)-(3.16) in the prioritization process appears to be a simple Quadratic Program with linear constraints that can be solved efficiently.

Note that when only systems of linear equalities are considered, with the additional final requirement of choosing x^* with a minimal norm, the prioritization process (3.14)-(3.16) boils down to the classical problem 2.19.

3.4.3 Algorithms

The proposed Algorithm consists in processing the priority levels from highest to lowest and solving at every level the corresponding Quadratic Program. The representation of the sets S_k by systems of linear equalities and inequalities is efficiently updated then by direct application of the properties (3.7) and (3.10).

It is naturally possible to optimize additional criteria over the final set of solutions. For instance, one might be interested in the solution with minimal norm, or in the solution that maximizes the distance to the boundaries of the optimal set, etc.



(a) Priority does not matter, the prioritized linear systems are compatible and the optimal sets is the intersections of their respective optimal sets.

(b) Equality has priority over inequality. The optimal set satisfies all possible inequalities while minimizing distance to the unfeasible ones.

(c) Inequality has priority over equality. The optimal set minimizes the distance to the equality's solution set.

Figure 3.3: Illustration of the optimal sets for prioritization problems involving both linear equality and inequality systems.

Algorithm 4 Solve prioritized linear systems

Initialize the system of equalities $\bar{A}_0$, $\bar{b}_0$ to empty.
Initialize the system of inequalities $\bar{C}_0$, $\bar{d}_0$ to empty.

for $k = 0$ to $p-1$ **do**

  Solve the Quadratic Program (3.15)-(3.16) to obtain S_{k+1}.

  $\bar{A}_{k+1} \leftarrow \begin{bmatrix} \bar{A}_k \\ A_k \end{bmatrix}, \bar{b}_{k+1} \leftarrow \begin{bmatrix} \bar{b}_k \\ A_k x_k^* \end{bmatrix}.$

  $\bar{C}_{k+1} \leftarrow \bar{C}_k, \bar{d}_{k+1} \leftarrow \bar{d}_k.$

  for all c_k^j in C_k **do**
    if $c_k^j x_k^* \leq d_k^j$ **then**

      $\bar{C}_{k+1} \leftarrow \begin{bmatrix} \bar{C}_{k+1} \\ c_k^j \end{bmatrix}, \bar{d}_{k+1} \leftarrow \begin{bmatrix} \bar{d}_{k+1} \\ d_k^j \end{bmatrix}.$

    else

      $\bar{A}_{k+1} \leftarrow \begin{bmatrix} \bar{A}_{k+1} \\ c_k^j \end{bmatrix}, \bar{b}_{k+1} \leftarrow \begin{bmatrix} \bar{b}_{k+1} \\ c_k^j x_k^* \end{bmatrix}.$

    end if
  end for
end for

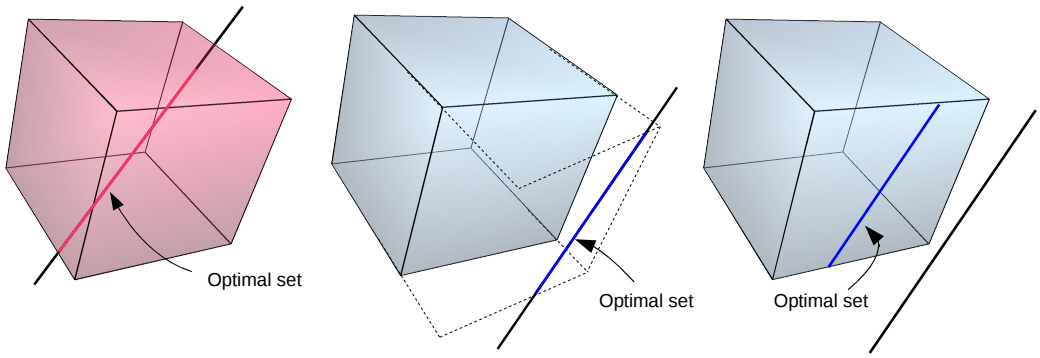


Figure 3.4: Equivalent of figure 3.3 in a 3D space.

Note that a similar algorithm has already been described by [Wolf 1996], but in the setting of Constraint Programming on discrete variables: the structure and the logic are similar, but the inner workings are very different, especially the theoretical analysis of section 3.3.

For an equality task $f(q) = 0$, the value of the task was implicitly defined in previous chapter as the convex function:

$$q \longmapsto \|f(q)\| \tag{3.17}$$

For an inequality task $g(q) \leq 0$, we use the convex value function:

$$q \longmapsto \sqrt{\sum_j max(0, g^j(q))^2} \tag{3.18}$$

with g^j being the j-th inequality in g. This is perhaps better known as an exterior penalty function[Fiacco and McCormick 1987] in the context of non-linearly constrained optimization. In case of a mixture of equality and inequality tasks, for instance $\{f(q) = 0, g(q) \leq 0\}$, the sum of all value tasks is taken:

$$q \longmapsto \|f(q)\| + \sqrt{\sum_j max(0, g^j(q))^2} \tag{3.19}$$

Apart from this modification of the task values, the iterative process (see Algorithm 5) that solves k prioritized equality or inequality tasks stays the same as in Algorithm 3.

3.4.4 Dealing with singularities

The proposed algorithm inherits the singularity issues from the classical algorithm based on pseudo-inversion. Recall that the pseudo-inverse could be regulated or thresholded to ensure that the joint updates do not diverge near singularity. In this new framework, we can take advantage of the same solutions. Thus, at every priority stage of Algorithm 4, instead of solving the problem (3.15)-(3.16) we may choose

Algorithm 5 Solve k prioritized tasks (equality, inequality or mixture)

Define ε_p the minimum task value progression required to continue
Call p the measure of task value progression
repeat

  for i from 1 to k **do**
    Calculate value of task i: $T_i^0(q)$ using (3.17), (3.18) or (3.19)
  end for

  Solve prioritized linear systems [Algorithm 4]
  Update configuration $q \leftarrow q + \delta q$

  for i from 1 to k **do**
    Calculate value of task i: $T_i^1(q)$
    Evaluate progression $p = T_i^1(q) - T_i^0(q)$
    if $p > \varepsilon_p$ **then**
      break.
    end if
  end for

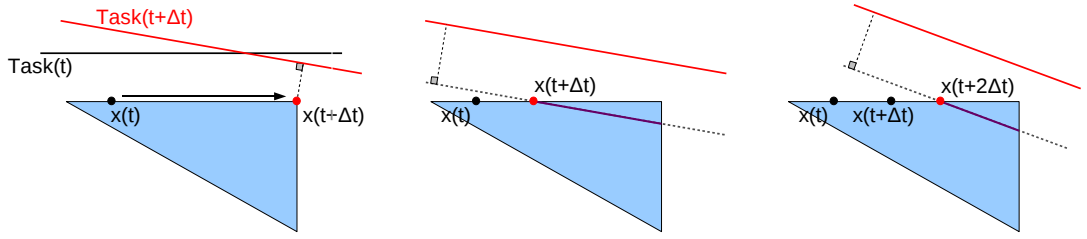
until $p < \varepsilon_p$

to solve:

$$S_{k+1} = \underset{x \in S_k, w \in \Re^m}{\text{Arg min}} \frac{1}{2}\|A_k x - b_k\|^2 + \frac{1}{2}\|w\|^2 + \lambda^2\|x\|^2 \tag{3.20}$$

$$\text{with } C_k x - w \leq d_k \tag{3.21}$$

where λ is a scalar. The following steps in Algorithm 4 are not modified. In figure (3.5) we illustrate the effect of the proposed method in a 2D space.



(a) The red line represents the solution set of a linear equality system that is singular with respect to feasible set, represented as a blue triangle. Slight variation of the red line may produce a large difference in the optimal point x.

(b) Damping with factor $\lambda^2\|x\|^2$ reduces the norm of x and augments the dimension of the optimal set (from the single red point in (a) to the purple segment).

(c) Next time iteration sees the optimal point x and the optimal set change continuously, provided the task is continuous.

Figure 3.5: Illustration of the effect of damping in problem (3.20)-(3.21) across successive time steps.

3.5 Conclusion

We have introduced inequality tasks as generalization to both inequality constraints and equality tasks. We have defined a unified algorithm for linear equality and inequality system prioritization. This algorithm can serve as a basis for numerical resolution of equality and inequality tasks regardless of the type of parameters: joint velocities, torques, accelerations, etc.

For humanoid robot, the physical constrains that express as inequality constraints, such as joint limits, non-collision with obstacles, static equilibrium, can be accounted for in a robust manner within our framework. As for the concept of inequality tasks, it will permit a more relaxed specification of tasks and we will give several examples thereof in the following chapter.

4 Illustration

4.1 A humanoid robot

The humanoid robot HRP-2 was used for our work. This robot can be modeled as a kinematic tree of links, connected by joints. This kinematic model has 28 revolute joints of freedom and extra ones at each hand to allow one-dimensional opening and closing. A link i in this model represents a solid body in the robot with its shape, its mass m_i and its inertial matrix I_i. A joint is the abstract object representing a single degree freedom between a parent and one child link. The joint is characterized by a parameter q_k that determines the position and orientation of a child link $k+1$ with respect to the parent link k.

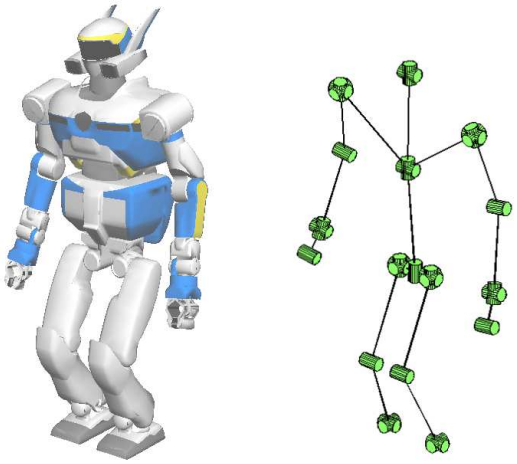


Figure 4.1: The humanoid robot HRP-2.

Call:

- F a frame attached to the workspace
- F_k the frame attached to link k.
- O_k the center of frame F_i.

For a parent link k and its child link $k+1$, define:

- (O_k, a_k) the axis of rotation of joint k.
- a_k the unitary vector defining the positive rotation of joint k, expressed in F_k.
- b_k the constant vector $\overrightarrow{O_k O_{k+1}}$ separating frames F_k and F_{k+1}, expressed in F_k.
- $Rot(w, \alpha)$ the rotation matrix representing a rotation around vector w by angle α.
- p_k the position of point O_k in frame F
- R_k the rotation matrix defining the frame F_k in frame F
- w_k the rotation vector equivalent to R_k, such that $R_k = Rot(\frac{w_k}{\|w_k\|}, \|w_k\|)$

The position and orientation of link i child of link k can be calculated as:

$$\begin{aligned} p_{k+1} &= p_k + R_k b_k \\ R_{k+1} &= R_k + R_k Rot(R_k a_k, q_k) \end{aligned}$$

With these formulas, any point in the kinematic structure can be computed recursively from the root node of the tree of links, which can be any link. The humanoid robot having complete motion freedom in space, the root node is considered linked to the environment by a 6-degree-of-freedom joint authorizing any transformation. There are applications that require interruption of contact between the robot's feet and the ground, such as jumping and running. In such cases, the position and the rotation of every link in the robot is known only if the root body's are known. In the scope of this work, the robot is permanently in non-slippery contact with a horizontal ground by one or both of its feet. Therefore, defining a root node other that a support foot was redundant given the knowledge of the foot's position and orientation in the workspace.

When it is not possible to select the root node for a given reason (e.g constraint of implementation, etc) Algorithm 6 can be used to compute the position and orientation jacobian of a frame with respect to another, in the kinematic structure. Given that a support foot is considered in non-slippery contact with the workspace, computing a jacobian with respect to the frame of a support foot is equivalent to computing the jacobian with respect to F, the frame attached to the workspace. The jacobians used for the differentiation of constraints and tasks presented in the next section rely on such computations.

Algorithm 6 Compute the jacobian of p_i and w_i in frame F with respect to frame F_k

1: Find the chain of joints A from root link to link i
2: Find the chain of joints B from root link to link k
3: $C = A \cup B - A \cap B$ is the set of joints between links i and k
4: **for all** $c \in C$ **do**
5: Calculate the rotation vector a of joint c in frame F according to case:
6: **if** $c \in A$ **then**
7: $a = R_c a_c$
8: **else**
9: $a = -R_c a_c$
10: **end if**
11: Position part:
12: $\frac{\partial p_i}{\partial q_i} = a \times (p_i - O_c)$
13: Orientation part:
14: $\frac{\partial w_i}{\partial q_i} = a$
15: **end for**

4.2 Constraints and tasks

Posture and limits

The simplest expressions of a kinematic tasks are:

$$q - q_{ref} = 0 \tag{4.1}$$

or

$$q - q_{ref} \leq 0 \tag{4.2}$$

with q_{ref} being a reference vector in $\Re^n$. The posture task in equality type are usually placed at the lowest priority level in the stack of prioritized tasks, acting like a constant attractor for a preferred configuration. In inequality, it can be used to define a range of preferred postures, yet the most common use is as constraint to observe joint limits.

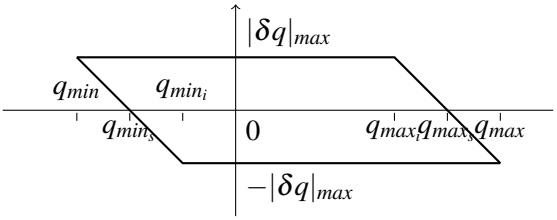


Figure 4.2: Bounding joint updates to enforce joint limits.

To enforce the joint limits described by the double inequality, $q_{min} \leq q \leq q_{max}$, we place bounds on the absolute value of the joint updates such that the update is gradually decreased to 0 when the joint parameter value is almost to its limit. We implement this approach by setting the differential system of

inequalities as a prior constraint for any prioritized optimization problem:

$$\delta q \leq \begin{cases} |\delta q|_{max} \frac{q - q_{max_s}}{q_{max_i} - q_{max_s}} & \text{if } q > q_{max_i} \\ |\delta q|_{max} & \text{otherwise} \end{cases} \tag{4.3}$$

$$-\delta q \leq \begin{cases} |\delta q|_{max} \frac{q - q_{min_s}}{q_{min_i} - q_{min_s}} & \text{if } q < q_{min_i} \\ |\delta q|_{max} & \text{otherwise} \end{cases} \tag{4.4}$$

where q_{min_i}, q_{max_i}, q_{min_s}, q_{max_s} and $|\delta q|_{amx}$ are as implicitly defined by figure 4.2.

Position and orientation

The most common tasks and constraints involve positions and orientations of bodies in the robot. To express an inequality task on the position and the orientation of a frame F_k, we write:

$$\begin{aligned} p_k &= p_{ref} \\ R_k &= R_{ref} \end{aligned}$$

where p_{ref} and R_{ref} are the desired position and rotation matrix for F_k. Call w_{gap} the rotation vector such that $R_k^T R_{ref} = Rot(\frac{w_{gap}}{\|w_{gap}\|}, \|w_{gap}\|)$. The linear equality system to be fulfilled write as:

$$J_k \delta q = \lambda \begin{bmatrix} p_{ref} - p_k \\ R_k w_{gap} \end{bmatrix} \tag{4.5}$$

where J is the full jacobian of the non-root foot's frame calculated for instance with Algorithm 6. In case we are considering a constraint to be permanently constrained, λ is taken equal to 1.

Plane

We are interested in the position of a point $P(q)$ on the kinematic structure. If we desire this point to be on a certain plane of the workspace, we write:

$$\langle u_n | P(q) \rangle = c \tag{4.6}$$

where u_n and c are the normal vector and the scalar defining the plane. For the numerical resolution, the linear system to be considered is:

$$\langle u_n | J_P \rangle \delta q = -\lambda (\langle u_n | P \rangle - c) \tag{4.7}$$

with $J_P \in \Re^{3 \times n}$ being the jacobian of the position of $P(q)$ expressed in the same frame as P and u_n.

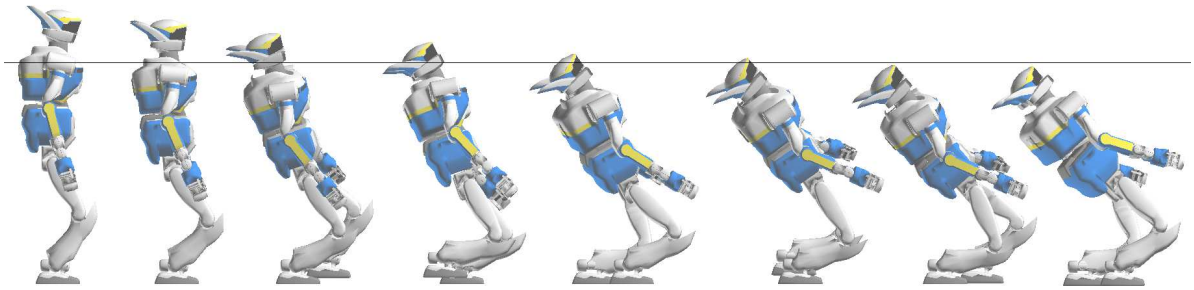


Figure 4.3: While the robot is walking, an inequality task is set such that a point in the head of the robot should go below a horizontal plane (the black line).

Parallelism

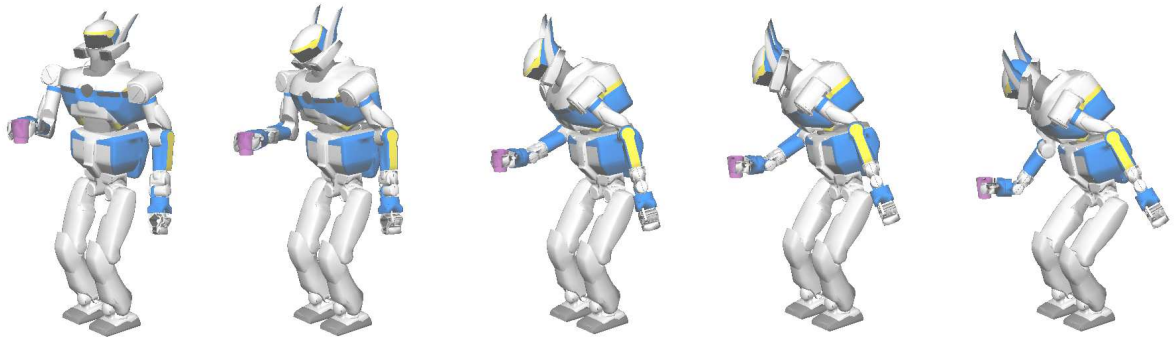


Figure 4.4: The glass is maintained vertical by setting a constraint on hand axis to remain parallel to the workspace's $\vec{z}$ vector.

We are interested in the orientation of an axis (O_i, u_l) attached to a body in the kinematic structure. We would like to align this axis with a reference axis (O, u_r) in workspace, thus we write:

$$u_r \times u_l = 0 \tag{4.8}$$

We differentiate to construct the linear system for numerical resolution:

$$u_r \times (J_{\omega_i} \delta q \times u_l) = -\lambda u_r \times u_l \tag{4.9}$$

where J_{ω_i} designates the jacobian of the rotation speed ω_i of link i. Replacing cross products with equivalent skew-symmetric matrices the above system can be re-written as:

$$[u_r]_\times [u_l]_\times J_{\omega_i} \delta q = \lambda u_r \times u_l \tag{4.10}$$

This task can be used to maintain a plate or a glass vertical in the hands of a humanoid robot (see figure 4.4). We often use it to maintain the robot straight.

Cone of directions

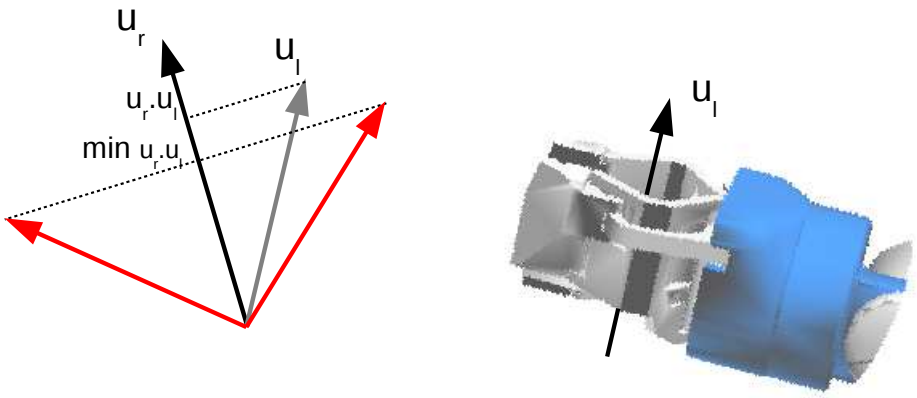


Figure 4.5: Setting a minimum value for $u_l.u_r$ constrains u_l to stay within a maximum angle around u_r.

A half-empty bottle could be held a little inclined without risk of spilling. To define such a constraint on any link i of the robot, we considered a vector u_l attached to the link and a vector u_r representing the reference direction from which we allow u_l to deviate (see figure 4.5). The corresponding inequality task writes as:

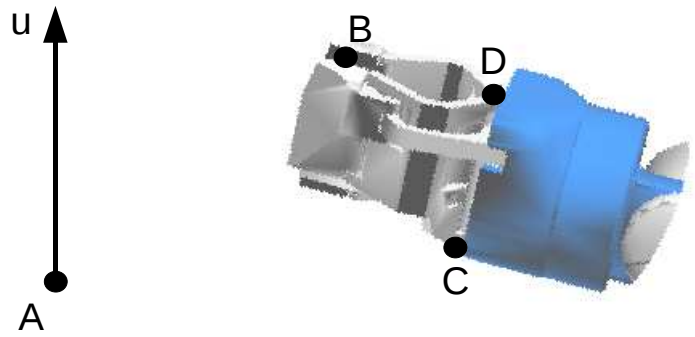
$$u_r.u_l - c > 0 \tag{4.11}$$

where c is the minimum cosine allowed between u_r and u_l, giving the differential inequality system:

$$-u_r.[u_l]_\times J_{\omega_i} \delta q \leq \lambda(u_r.u_l - c) \tag{4.12}$$

where J_{ω_i} designates the jacobian of the rotation speed ω_i of link i, $[u_l]_x$ the skew-symmetric matrix representing a prior cross product by u_l.

Coplanar line segments



Consider HRP-2's hand having to grasp a cup. The sagittal plane defined by the hand clamps has to coincide with the axis of the cup. We have this if two non parallel line segments of the hands' plane are coplanar with the axis of the cup.

Let $(A, \vec{u})$ define a reference axis and let B, C and D be three points defining the constrained plane on a a robot link i (figure 4.2). We express the task as:

$$\begin{array}{rcl} (u \times \overrightarrow{AB}) \times (u \times \overrightarrow{AC}) & = & 0 \\ (u \times \overrightarrow{AB}) \times (u \times \overrightarrow{AD}) & = & 0 \end{array} \tag{4.13}$$

Letting J_B, J_C and J_D be the jacobians of the position of points B, C and D on link i, the differential system is:

$$\begin{array}{rcl} \{([\overrightarrow{AC} \times u]_\times [u]_\times) J_B - [\overrightarrow{AB} \times u]_\times [u]_\times) J_C\} \delta q & = & -\lambda (u \times \overrightarrow{AB}) \times (u \times \overrightarrow{AC}) \\ \{([\overrightarrow{AD} \times u]_\times [u]_\times) J_B - [\overrightarrow{AB} \times u]_\times [u]_\times) J_D\} \delta q & = & -\lambda (u \times \overrightarrow{AB}) \times (u \times \overrightarrow{AD}) \end{array} \tag{4.14}$$

Gaze direction

A variation of the previous task is applied to control the optical axis of a vision system attached to the structure. Let $O(q)$ be a point on the optical axis, u_g a unitary vector along the optical axis oriented towards the vision field, T a fixed point representing the target of focus in the workspace and u_t the unitary vector defined by $u_t = \frac{\overrightarrow{OT}}{\|\overrightarrow{OT}\|}$. Writing the following equation, we require the vision system to focus on point T:

$$\overrightarrow{OT} \times u_g = 0 \tag{4.15}$$

After differentiation the linear system for numerical resolution is:

$$([u_g]_\times J_O - [u_t]_\times [u_g]_\times J_\omega)\, \delta q = -\lambda u_t \times u_g \tag{4.16}$$

Suppose that instead of defining a focus point on which the optical axis must be constantly aligned, we prefer to specify a region of focus described by a solid angle. In this case, we rewrite the task as two inequalities, as seen in the previous subsection.

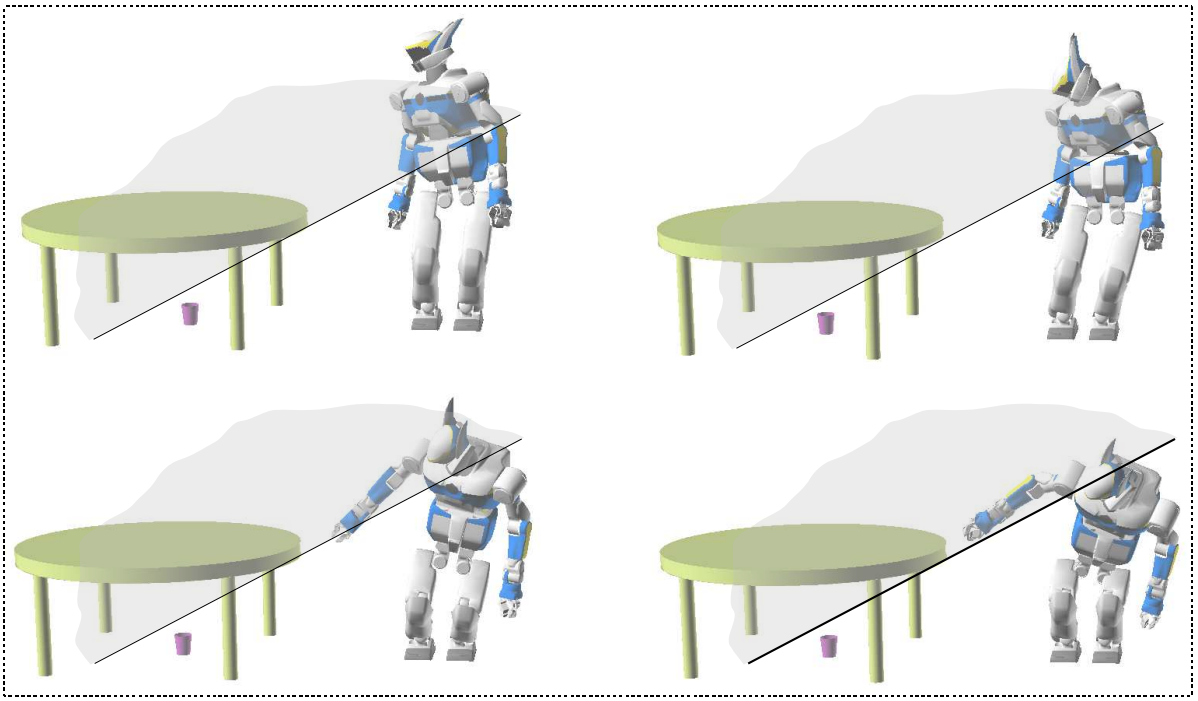
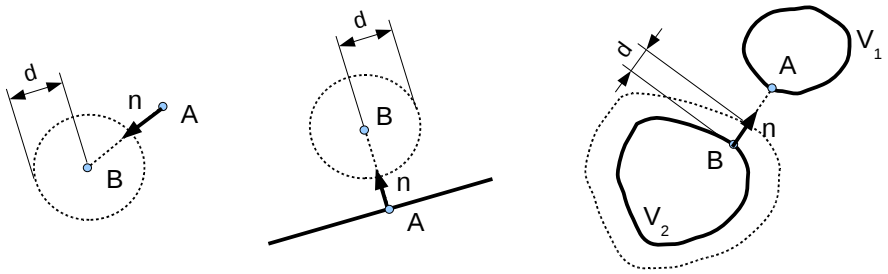


Figure 4.6: Two simultaneous tasks: a gaze equality task to look at a point under the table and a plane inequality task to draw the head out of the occlusion area (in transparent gray).

Collision avoidance



Forbidding collision between two elements is preventing the shortest distance between them to become null. Between a point A and a point B, between a point and a line or a line segment, or between two strictly-convex volumes V_1 and V_2, there is unique pair of points realizing the minimal distance (figure 4.2). In these cases, collision is avoided if we observe the constraint:

$$\langle \overrightarrow{AB} | \mathbf{n} \rangle - d \geq 0 \tag{4.17}$$

where d is the minimal distance to be kept between points A and B, and $\mathbf{n} = \frac{\overrightarrow{AB}}{\|\overrightarrow{AB}\|}$.

One may observe that the relative rotation between the objects in second and third case may still be the cause of collision (if, for instance, object V_1 rotates around point A by $\frac{\pi}{2}$ in the next time sample),

however, since the absolute rotation speed is bounded, the maximal displacement of any point on either of the elements is also bounded, therefore the distance d could always be chosen such that violation of the minimal distance d remains very small.

The differential system following from constraint 4.17 is thus:

$$-\langle \delta(\overrightarrow{AB})|\mathbf{n}\rangle \leq \lambda(\langle \overrightarrow{AB}|\mathbf{n}\rangle - d) \tag{4.18}$$

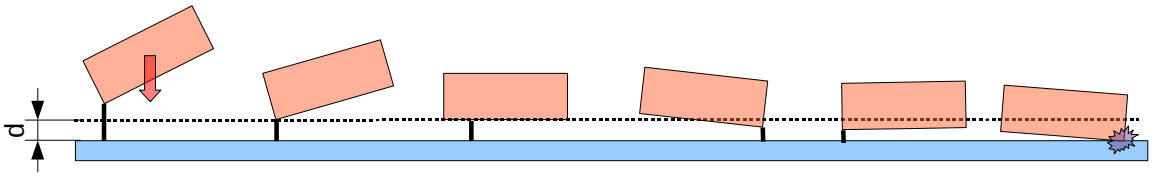


Figure 4.7: Inspired from [Kanehiro et al. 2008]. Constraining a single pair of points realizing the minimum distance (represented by vertical black line) between non-strictly convex shapes does not prevent collision.

This method was applied by [Faverjon and Tournassoud 1987] on the the pairs of points realizing the shortest distance between a mobile robot and the obstacles around, bounded by strictly-convex shapes. A recent work by [Kanehiro et al. 2008] focused on the general case where the objects are non strictly-convex polyhedra. They showed how a collision may occur when only a single pair of points realizing the minimal distance is constrained between a segment and a line in a plane (see figure 4.7). The method they propose is an extension of the work by [Faverjon and Tournassoud 1987] where they analyze the pairs of points to be constrained for all configuration cases between faces, edges and vertices of polyhedra. For instance, in the example of the segment and the line in a plane, their analysis leads to constrain the extremities of the segment as suggested in figure 4.8.

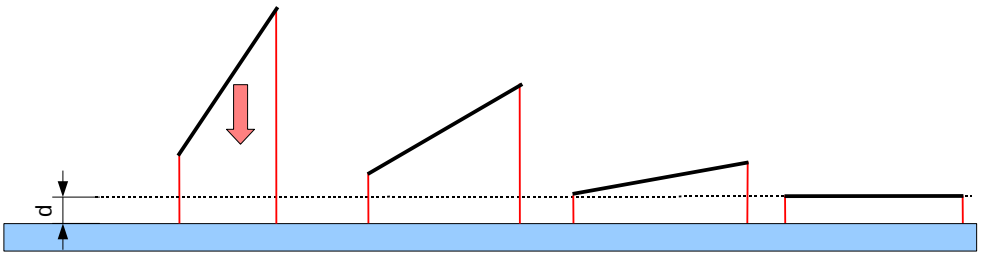


Figure 4.8: Inspired from [Kanehiro et al. 2008]. At least two linear velocity constraints are needed to prevent collision between a line segment and a plane

The algorithm they propose permits accurate collision avoidance but can be too costly if the geometrical model of the robot is complex. We have consequently designed a simplified geometry for the robot where every link is wrapped in a cylinder capped with half spheres. The interesting aspect of this particular geometrical shape is that it represents a line segment around which a minimal distance d,

corresponding to the radius of the cylinder and the spheres, need be observed (figures 4.9 and 4.10). All that remains is to follow the same analysis conducted by [Kanehiro et al. 2008] to determine the pairs of points that need be constrained between line segments in 3D space.

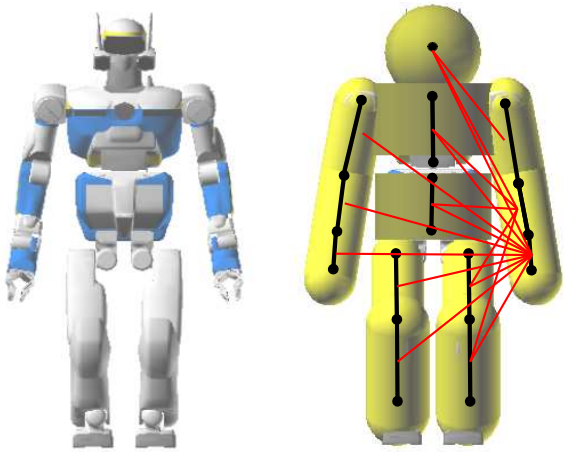


Figure 4.9: Setting simple constraints for self-collision avoidance: the bodies of the robot are wrapped in cylinders and spheres. The red lines are examples of pairs that need to be watched for collisions.

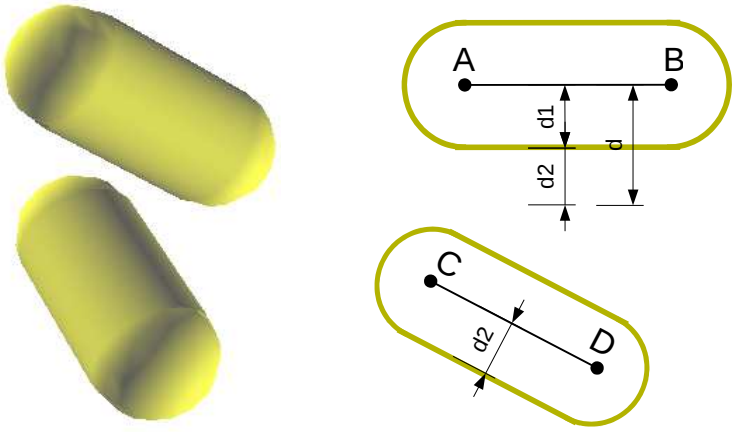


Figure 4.10: The approximation of a link with a cylinder and half-spherical caps is equivalent to an approximation by a line segment that must not be approached by less than the radius of the cylinder.

Let us expose the solution for line segments in a plane, since it will be very close to the 3D case. Figure 4.11 shows the three configuration cases that demand different treatment. The cases are defined according to intersection of segment $[CD]$ with two type of regions defined with respect to segment $[AB]$: the middle region where orthogonal projections on the line (AB) fall on the segment $[AB]$, and the complementary regions containing only points A and B. d represents the minimal authorized distance

between the line segments. In the first case, the forbidden region (half sphere) is strictly convex, therefore a single pair of points is enough to maintain $[CD]$ out. In the second case, $[CD]$ completely inside the middle region, thus the linear velocities of its extremities are constrained. In the third case, $[CD]$ is between two adjacent regions. It is convenient to think of it as two separate line segments, each in a region enabling us to apply the constrains found for first and second case, yielding a total of three constrains.

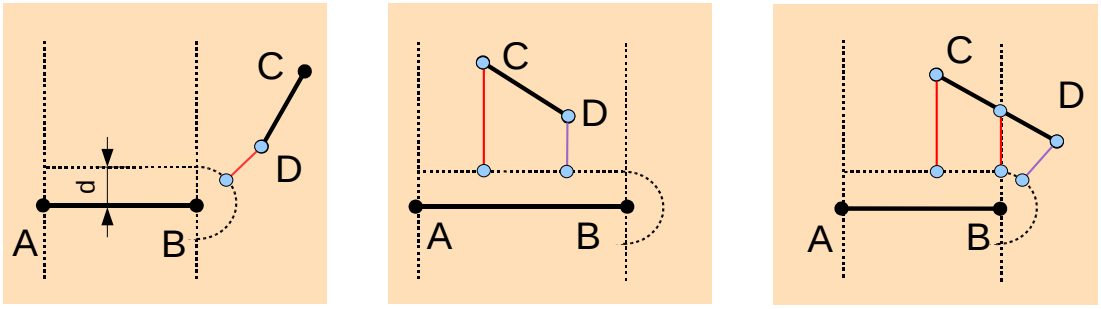


Figure 4.11: Pairs of points to constrain to maintain a minimal distance between two plane segments.

For the 3D case that is in our interest, the region analysis holds, but we have to take one additional case into account, depicted in figure 4.12. It occurs when the minimal distance between the line segments lies in the middle region, and is realized by a pair of points excluding the extremities. To obtain the distance and the pair of points realizing it we can write a point X on $[AB]$ as:

$$X = A + \lambda_1 \overrightarrow{AB} \quad \lambda_1 \in [0,1]$$

and a point Y in $[CD]$ as:

$$Y = C + \lambda_2 \overrightarrow{CD} \quad \lambda_2 \in [0,1]$$

and solve the optimization problem:

$$\left\{ \min_{(\lambda_1,\lambda_2)\in[0,1]^2} \quad \|Y - X\|^2 \right.$$

We have to pay attention to the fact that we project the points on the surface defined by the minimal distance d, whereas we projected on a line segment or on a circle in the 2D case.

We have now a simplified model of our robot and we can write the constraints that ensure self-collision avoidance for any configuration. For HRP-2, there was no need to constrain all pairs of links since some would never enter in collision either because of the geometry or because of the settings of use.

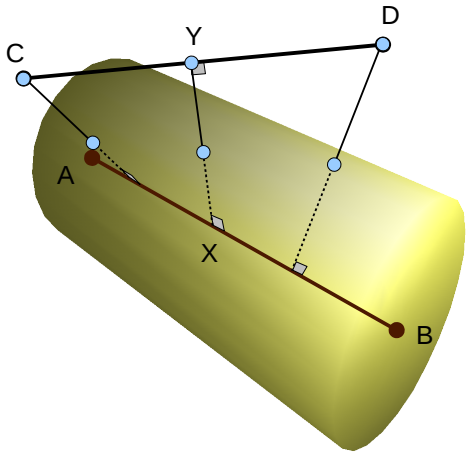


Figure 4.12: Extra case showing pair of points to constrain between two line segments in 3D space

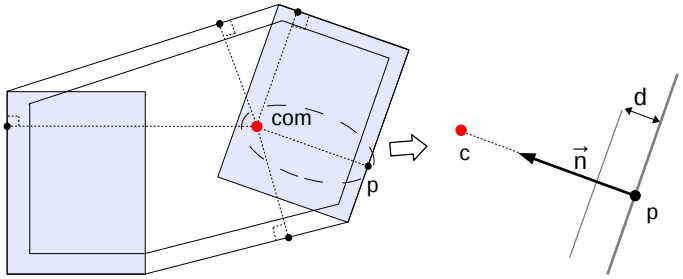


Figure 4.13: The velocity of approach of the center of mass projection to the edges of the support polygon is damped.

Static equilibrium

The static equilibrium for a humanoid robot standing on a horizontal ground is verified when the projection of the center of mass of the robot is inside its support polygon.

Call C the projection of the center of mass on the ground. The constraint that keeps C inside the support polygon is composed of a variable number of the linear differential inequalities. One linear inequality constraint is added when an edge of the support polygon is directly facing C (see figure 4.13). We control the maximal velocity of C towards the polygon edges by observing the following inequality:

$$-\langle \delta(\overrightarrow{PC})|\vec{n}\rangle \leq \lambda(\langle \overrightarrow{PC}|\vec{n}\rangle - d) \tag{4.19}$$

where $\vec{n}$ is the normal vector to the considered edge and d is the minimal authorized distance. The support polygon being fixed when the robot is standing on the ground, the above inequality boils down to:

$$-\langle J_C|\vec{n}\rangle \delta q \leq \lambda(\langle \overrightarrow{PC}|\vec{n}\rangle - d) \tag{4.20}$$

where J_C is the jacobian of C with respect to the joints configuration.

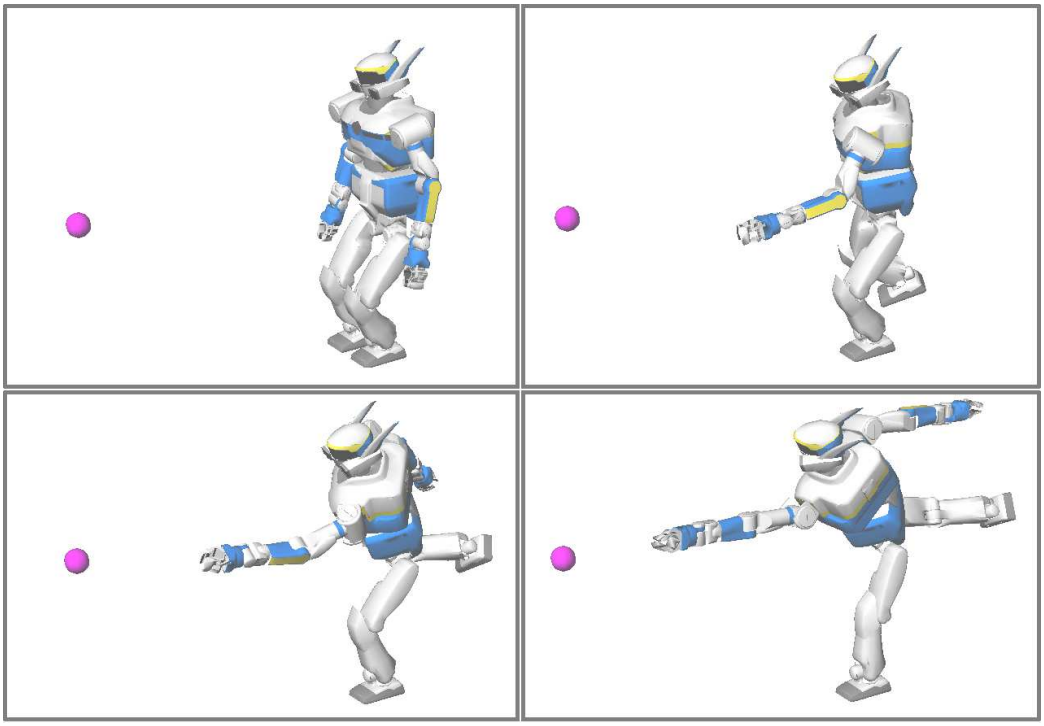


Figure 4.14: The projection of center of mass is constrained inside the left foot while the left hand tries to reach the front target. The right limbs are not controlled, their motion is caused by their influence on the position of the center of mass.

4.3 Prioritization

We have applied the proposed algorithm seen in section 3.4.3 to plan local motions for the humanoid robot HRP-2[Kaneko et al. 2004]. We show in the following examples the ability of our algorithm to treat any order of priority with both equality and inequality tasks.

The permanent set of constraints

Any task that we require from the robot is subject to a set of constraints that we list here:

- Joint limits
- Static equilibrium
- Self-collision and obstacle avoidance
- Fixed position and orientation of feet

We have already presented the form of linear systems applied to enforce joint limits ((4.3), (4.4)) static equilibrium (4.19), collision avoidance (4.17) and position and orientation of a body (4.5). For the

kinematic model of our humanoid robot, we defined one support foot as the root node of the tree due to its constant fixed contact with the environment (see section 4.1). Therefore, the fourth constraint in the list is satisfied by constraining the non-support foot only. As for collision avoidance constraints, for the following scenarios, we did not resort to the simplified geometry presented above, we kept the full model of the robot. The computation of distances between polyhedra was done with the motion planning software Kineo[Laumond 2006].

Priority under inequality constraints

In this example, we illustrate the utility of prioritizing equality tasks after specification of inequality constraints. The goal of the motion is to reach a ball underneath an object (blue polyhedron in figure 4.3) while looking at it. Here is the stack of tasks sorted in decreasing priority:

1. Permanent set of constraints
2. Reach for the ball
3. Look at the ball

For the reaching we specified a three-dimensional position task on the center of the left hand. The gaze task was defined as in section 4.2. For both tasks, we added a simultaneous optimization criterion as explained in section 3.4.4 to avoid large updates due to conflicting tasks.

In the resulting motion, the gaze task could be maintained until the robot's head came close to the border of the table. When simultaneous looking and reaching became infeasible, the specified priorities made the robot continue the reaching while its gaze direction drifted off the target. Task 2) was satisfied at the end of this motion (figure 4.15(b)).

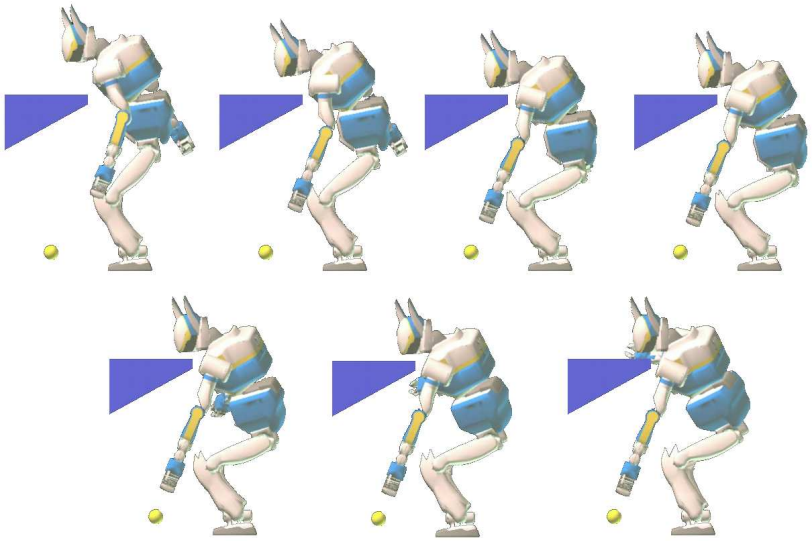
We tried to achieve the same goal while making the looking and the reaching tasks share the same priority. This time the robot was trapped in an intermediate configuration of the previous motion (frame 4.15(a)) where the looking task was maintained while the head over the table and the reaching hand stuck away from the ball.

One other method, presented by [Park et al. 1997] and applied to move a robotic arm, should also permit to solve this scenario. For the next one, there is no available resolution method to our knowledge.

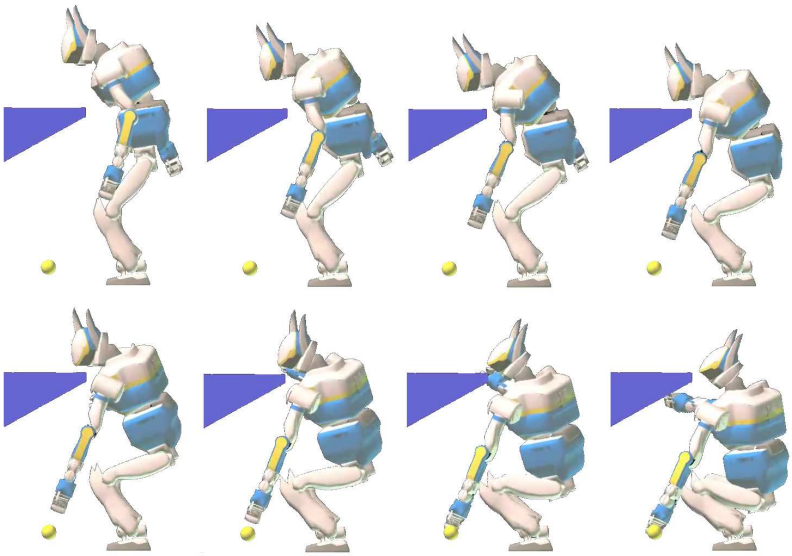
Inequalities below equality tasks

In this example we illustrate the ability of our prioritization algorithm to account for inequality tasks at low priority. The goal of the motion is to reach for a ball while avoiding, when possible, a region around the ball (represented in figure 4.3 by a green box). Here is the priority order for this motion:

1. Permanent set of constraints
2. Reach for the ball
3. Look at the ball
4. Avoid the green box



(a)



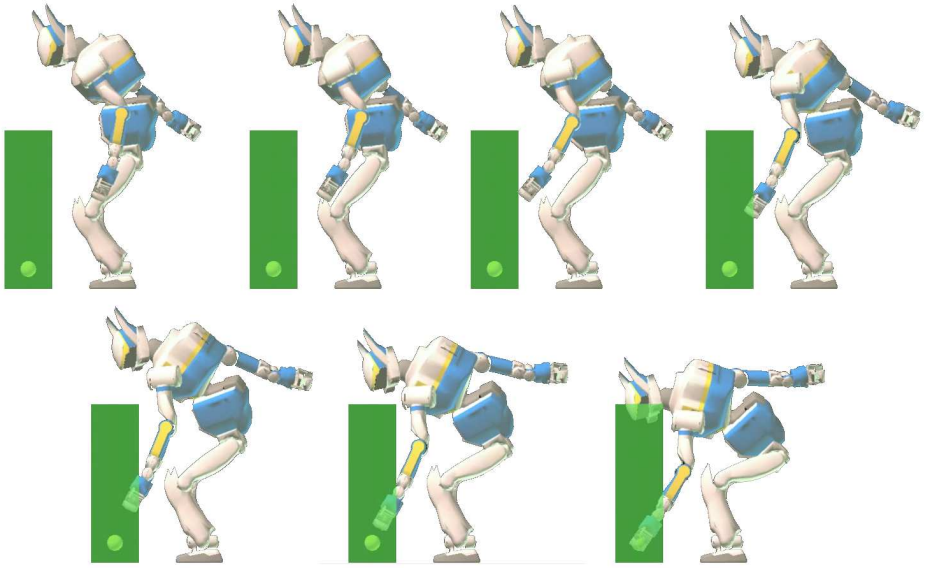
(b)

Figure 4.15: Under collision avoidance constraints (priority 1), the robot had to reach for the ball (priority 2) and look at it (priority 3). Figure (a) represents the classical case where all equality tasks are solved with the same equality under inequality constraints, figure (b) shows the result for prioritized equality tasks under inequality constraints allowed by our algorithm. Towards the end of the motion, the gaze task could not be maintained because of the hand pulling towards the ball and the presence of the table. The strict enforcement of priority enabled the reaching task to prevail and succeed.

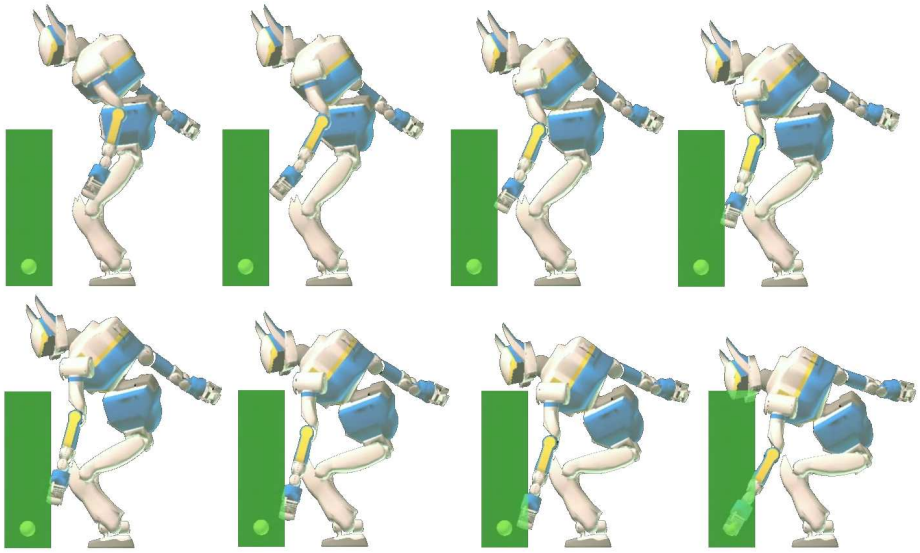
The idea behind placing the tall box on the ball is to guide the hand out of the vision field to avoid the occlusion of the ball. In a more rigorous but less simple implementation of this scenario, one would consider the vision cone linking the robot's head to the ball. For the illustration of the method, however, the green box suffices.

For the box avoidance task, four points in the body of the hand were checked for collision. The reaching task placed at priority 2) is different from the first example as it is one-dimensional only. This is done by allowing the hand to move on the orthogonal plane to the vector separating it from the target.

In the resulting motion, the hand was forced by task 4) to stay behind a face of the green box until it became incompatible with task 2). Then, the hand progressively entered the volume of the box and achieved its goal. As we expected, the inequality task 4) was maintained as long as possible and ended unsatisfied to the benefit of equality task 2).



(a)



(b)

Figure 4.16: This example illustrates how our algorithm could serve for target occlusion avoidance.The robot had to reach for the ball (priority 1), look at it (priority 2) while avoiding the green region (priority 3). Figure (a) shows the classical case without inequality task, figure (b) an inequality task is placed as last task for the hand to avoid the green box. Towards the end of the motion, reaching the task required the green region to be crossed, rendering the inequality unsatisfied. The higher priority task of reaching the ball was achieved.

5

Dynamic walk and whole body motion

The human motion of walking is a repeated cycle comprised of two alternated phases:

- Single support phase: the body is supported by a single leg, the other is displaced to its new fall point.
- Double support phase: this is a shorter phase during which the pressure of the body on the ground is shifted from a support foot to the next.

The durations of these phases greatly depend on the subject (skeletal structure, weight, clothing, etc...) and the setting (ground surface regularity, friction, slope, etc). So does the motion of the whole body during the full cycle.

In the previous chapter, we stated that the equilibrium of a humanoid robot could be maintained if the robot moves in quasi-static motion while the projection of its center of mass lies within its support polygon. A way to define quasi-static motion is to view it as the motion that can be "interrupted" and "resumed" at any time without compromising the equilibrium of the robot. Such motion can be easily computed to make the robot walk over planned footprints: first, shift the center of mass slowly so that its projection is under the first support footprint, then move the non-support leg in a slow and regular trajectory to finally land on the new footprint, the rest of the walking is the repetition of these two phases. If we want the robot to step at a velocity that matches a human's, we have to apply high accelerations on the joints of the robot. With the inertia of the body and the induced dynamics, a well calculated motion can lead to a well controlled jump, an ill-calculated one may result in an unavoidable fall.

We studied a method proposed by [Kajita et al. 2003] to generate dynamic walking motion for a humanoid robot walking on a flat ground (section 5.1). We looked into integrating this method in the prioritized inverse kinematics framework described in section 2.4 and 3.4 and found an easy solution to this problem (section 5.1.4). A generic implementation was prepared for the robot HRP-2 and lead to

successful field experiments, described in section (5.2).

5.1 Dynamic walk generation

5.1.1 Dynamic constraints

We consider a biped robot modeled as a set of articulated rigid bodies. We denote for rigid body k the following quantities:

- x_k the position of center of mass
- R_k the rotation matrix
- ω_k the rotation speed
- I_k the inertia matrix
- m_k the inertia matrix

The dynamic wrench of the system of rigid bodies composing the robot is:

$$\begin{bmatrix} \dot{\boldsymbol{P}} \\ \dot{\boldsymbol{L}} \end{bmatrix} = \begin{bmatrix} \sum m_k \ddot{x}_k \\ \sum x_k \times m_k \ddot{x}_k + R_k I_k \dot{\omega}_k - R_k((I_k \times \omega_k) \times \omega_k) \end{bmatrix} \tag{5.1}$$

$\boldsymbol{P}$ is the linear momentum of the system and $\boldsymbol{L}$ its angular momentum around the frame of reference. The total wrench of forces acting on the system is denoted:

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \sum m_k \vec{g} + \sum f_i \\ \sum m_k x_k \times \vec{g} + \sum p_i \times f_i \end{bmatrix} \tag{5.2}$$

where f_i are the forces of contact at positions p_i on the robot.

The fundamental principle of dynamics stipulates that the dynamic wrench of the system of rigid bodies is equal to the total wrench of the forces acting on the system:

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{P}} \\ \dot{\boldsymbol{L}} \end{bmatrix} \tag{5.3}$$

This is a constraint that is always verified for any state $\{x_k, \dot{x}_k, \ddot{x}_k\}$ of the rigid body system and the contact forces f_i may change or cease in time.

For our setting, we are interested in a walk motion on a flat ground without contacts outside the bottom of the feet. Furthermore, the motion must not produce jumps, which means that the set of contact points between the bottom of the feet and the ground should never become empty. Finally, the contact must not be slippery. We label these conditions as ***walk conditions***. For these requirements the analysis conducted by [Wieber 2002] applies and we report the condition derived from equation (5.3) that must be satisfied:

$$\frac{mgx_G \times \vec{n} + \vec{n} \times \dot{\boldsymbol{L}} \times \vec{n}}{mg + \dot{\boldsymbol{P}}.\vec{n}} = \frac{\sum f_i.\vec{n} p_i \times \vec{n}}{\sum f_i.\vec{n}} \tag{5.4}$$

$\vec{n}$ being the normal of the horizontal ground, and x_G the position of the center of gravity of the system. The point whose coordinates are expressed by either of the equation sides is called the Zero Momentum Point (ZMP [Vukobratović et al. 1990]). As a matter of fact, the horizontal angular momentum around the ZMP is null. Equation (5.4) shows that the ZMP lies in the convex hull of all contact points p_i where the vertical component f_i is strictly positive. At any time t, the walk conditions are thus:

$$\begin{cases} \mathscr{C} : \{p_i \text{ such that } f_i.\vec{n} > 0\} \neq \emptyset & \textit{ground contacts maintained} \\ \text{ZMP } \in \text{ convex hull } (\mathscr{C}) & \textit{dynamics fundamental principle} \\ \dot{p}_i = \vec{0} \ \ \forall p_i \in \mathscr{C} & \textit{non slippery contact} \end{cases} \tag{5.5}$$

From the control point of view, these conditions must be added to other constraints on control variables, such as torque limits on joint motors, limits on force of impact of feet on the ground during stepping, constraints such as those we used for the illustrations of section 4.2, etc.

5.1.2 A model approximation

The ZMP criterion (5.4) may appear as a complex constraint when applied to an arbitrary system of rigid bodies with contacts on the ground. Vukobratovic et al. had the idea to plan a joint motion for lifting a leg and use the sagittal sway motion for the upper body so that the ZMP falls within the planned support polygon. Analytical solutions such as this cannot be applied easily to redundant robots, for which numerical approaches are applied [Kanehiro et al. 2008; Collette et al. 2008]. The algorithms proposed in these methods rely on minimization of objective functions corresponding to tasks in the operational space[Khatib 1987] while observing the laws of mechanics, such as the friction cone limit at contact points, the fundamental principle of dynamics, etc. We have the aim of exploring these methods while taking advantages of the strict prioritization of linear equality and inequality systems. However, we rely so far on a simplified model of the dynamics and an optimal controller that allows us to achieve a dynamic walk within the velocity-resolved inverse kinematics framework.

The simplified model was proposed by [Kajita et al. 2003]. The biped structure is approximated by a wheeled cart whose mass is equal to the robot's, moving on top of a mass-less table. The contact area between the table and the ground represents the support polygon of the robot, the cart represents the center of gravity of the robot (see figure 5.1). According to [Vukobratović et al. 1990], the distribution of the reaction force of the ground can be replaced by an equivalent reaction force $\vec{R}$ exerted on the zero momentum point. The other forces acting on the table are the weight of the cart $-mg\vec{n}$ and the tangent force exerted by the wheels on the top's surface, which is $-m\ddot{x}$. Therefore, letting z_h be the height of the center of mass (the wheeled cart), the dynamic angular momentum at the ZMP is:

$$\tau_y = (x - p_x)mg - z_h m\ddot{x}$$

which must be equal to 0. The same analysis is conducted on the y coordinate, yielding the position of

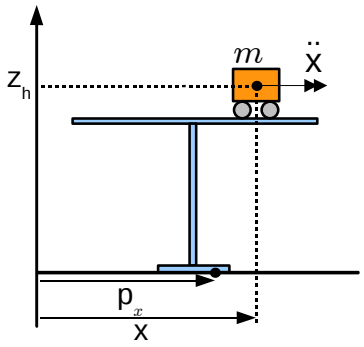


Figure 5.1: Cart-table approximation of the footstep dynamics. The cart represents the center of mass of the robot in free translation on the plane defined by the top of a mass-less table, representing the contact of the body with the ground.

the ZMP:

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} x - \frac{z_h}{g}\ddot{x} \\ y - \frac{z_h}{g}\ddot{y} \end{bmatrix} \tag{5.6}$$

When the ZMP is on an edge of the table's foot, the distribution of the ground reaction force is on that edge only, meaning that the table (and robot's foot) has started to topple over. Since this is a difficult situation to recover from, it would be best if the ZMP stayed close to the center of the support area.

Comparing with the general expression (5.4), the cart-table approximation cancels the the centrifugal and Coriolis dynamics of the system. This situation is never true since, even if the upper body is not rotated, the legs' rotations are not symmetrical during a footstep. The model is still practically efficient. The remaining question is how to control the ZMP to stay inside the support polygon.

5.1.3 An optimal controller

The equality (5.6) establishes a linear variation of the ZMP in the position of the center of mass and its second time derivative. As reported by [Kajita et al. 2003], such equality could be inverted by several means (Fourier transform, piece-wise polynomials fitting,etc), the objective being to plan a valid ZMP trajectory and control via the center of mass. otherwise expressed, the initial problem of controlling the full dynamic state to fulfill the dynamic constraints (5.4) is transformed into a problem of controlling the center of mass alone. Naturally, the approximation of the model does not allow us to account for full dynamic motions, thus restraining the freedom of motion.

To control the center of mass with respect to a planned ZMP trajectory, Kajita et al. suggest a method based on an optimal linear-quadratic control with an infinite preview horizon. We report here the method as they described. Introducing the scalar control variable:

$$u_x = \frac{d^3x}{dt}$$

the dynamics of the cart-table system along x axis write as:

$$\begin{aligned}\frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_x \\ p_x &= \begin{pmatrix} 1 & 0 & -\frac{z_h}{g} \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix}\end{aligned}$$

The corresponding discrete system for a time period T is described by:

$$\begin{aligned} X(k+1) &= A_k X(k) + b_k u_X(k) \\ P_X(k+1) &= CX(k) \end{aligned}$$

where:

$$\begin{aligned} X(k) &= \begin{pmatrix} x(kT) & \dot{x}(kT) & \ddot{x}(kT) \end{pmatrix} \\ u_X(k) &= u_X(kT) \\ p_X(k) &= p_X(kT) \\ A &= \begin{pmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \\ B &= \begin{pmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{pmatrix} \\ C &= \begin{pmatrix} 1 & 0 & -\frac{z_h}{g} \end{pmatrix} \end{aligned}$$

The quadratic cost to be minimized for this system is:

$$J(k) = \sum_{i=k}^{+\infty} Q_e e(i)^2 + \Delta x(i)^T Q_x \Delta x(i) + R\Delta u(i)^2$$

where:

$$\begin{aligned} \Delta u(i) &= u_X(i) - u_X(i-1) \\ \Delta x(i) &= X(i) - X(i-1) \\ \Delta e(i) &= p_x(i) - p_x^{ref}(i) \end{aligned}$$

p^{ref} denoting the reference ZMP trajectory, Q_x a symmetric non-negative definite matrix, Q_e and R strictly positive scalars.

The cost function $J(k)$ minimizes a weighed sum of three terms: the error of ZMP with respect to its reference trajectory, the variation of the state of the center of mass X and the variation of the control u_x. The optimal control obtained from the minimization of J is known:

$$u_X(k) = -G_x X(k) - G_e \sum_{i=0}^{k} e(i) - \sum_{j=0}^{+\infty} G_p(j) p^{ref}(j+k)$$

The same analysis and result is conducted to derive the control the control u_y of the second coordinate of the center of mass. The control. thus defined, takes into account the integral of past errors on the trajectory, the current state of the system and the trajectory of the ZMP to come. The number of terms due to this last part is infinite. Nonetheless, $G_p : k \mapsto G_p(kT)$ is a quickly-evanescent function (see figure 5.2), what allows us to take into account a finite number of gain coefficient, thus of future ZMP trajectory samples. The major advantage is the possibility to run such a controller of the ZMP online.

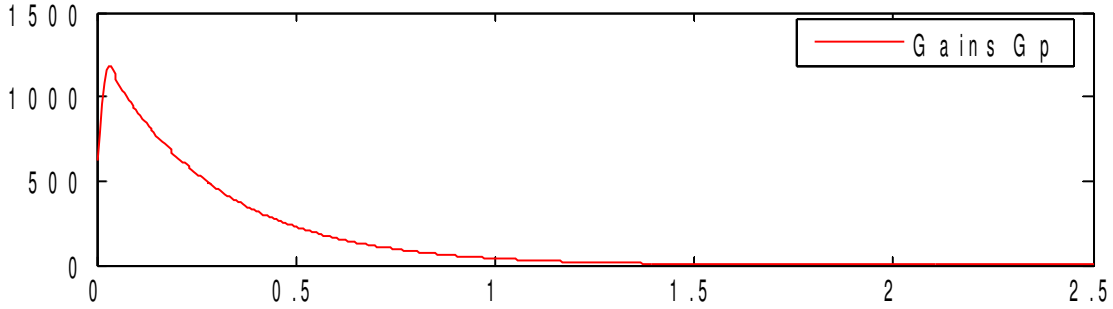


Figure 5.2: The G_p gains coefficients function of the time (for the robot HRP-2 with its center of mass at height $0.65m$). The coefficients beyond time 1.6s are relatively very small and can be omitted for the control of the center of mass without noticeable impact on the accuracy.

We show in figure 5.3 an example of spacial planning of the ZMP according to the planned footsteps. The reference trajectory planned on this path is given in figure 5.4 in dashed line. Notice that the trajectory is slightly smoothed between a single support phase (indicated by stable coordinates of the ZMP) and the following double support phase. We show in green line the trajectory of the center of mass calculated with the optimal controller. While the ZMP is piece-wise linear, the center of mass has a smooth oscillating trajectory. Figure 5.5 shows the spacial view of both trajectories.

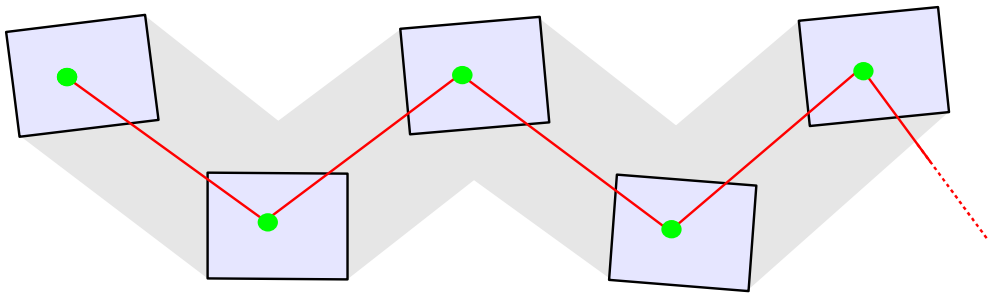


Figure 5.3: Planning the ZMP path within the support polygons. The green discs represent the fixed ZMP position during a single support phase, the red line represents the path traveled by the ZMp from one support foot to the other during a double support phase.

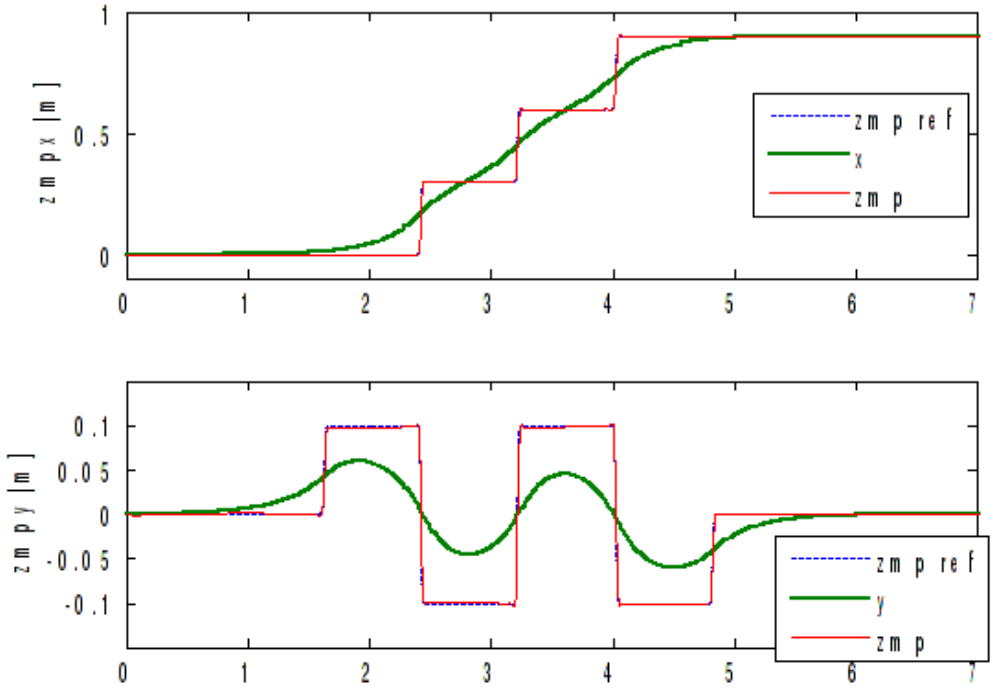


Figure 5.4: Reference and result trajectories of ZMP and control trajectory of the center of mass.

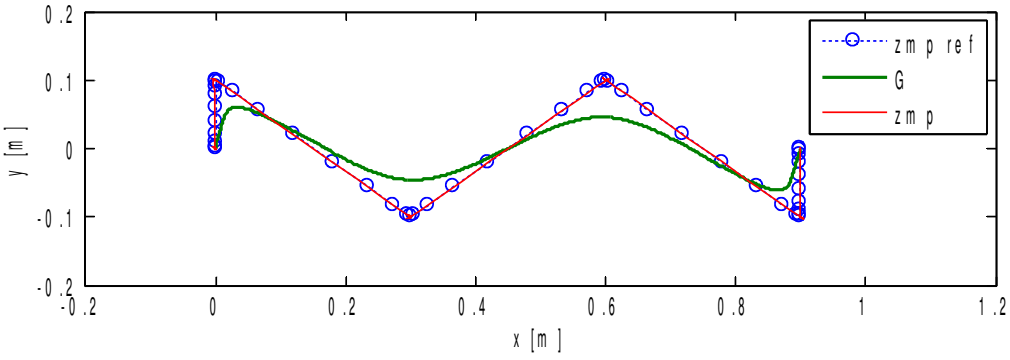


Figure 5.5: Path of the ZMP and center of mass during the walk.

5.1.4 Merger with prioritized inverse kinematics

One of our contributions[Yoshida et al. 2006] was to merge the dynamic walk generation framework described above with the prioritized inverse kinematics framework described in the first part of this document. The central idea is to place the tasks which are required for the dynamics walk in first priority since they involve the equilibrium of the robot and cannot be hindered by any other desired motions. The tasks to be taken into account are:

1. Position of the center of mass
2. Position and orientation of the non support foot

Let us recall that we constrain a single foot to the ground since the other is considered is non-slippery contact with the ground, as seen in section 4.1.

The first task is dictated directly by the preview controller: at time step kT, the control

$$\begin{pmatrix} u_X(kT) \\ u_Y(kT) \end{pmatrix}$$

is calculated from the planned samples

$$(p^{ref}(kT), ..., p^{ref}((k+N)T))$$

and is used to calculate the state of the center of mass

$$\begin{pmatrix} X((k+1)T) & Y((k+1)T) \end{pmatrix}$$

A task is built at time step k to bring the position of the center of mass to its desired position at time $k+1$:

$$\begin{pmatrix} x(q) \\ y(q) \end{pmatrix} = \begin{pmatrix} x((k+1)T) \\ y((k+1)T) \end{pmatrix}$$

The linear equality system derived from this task involves the computation of the jacobian of the center of mass[Sugihara et al. 2002].

The second task varies according to the walk phase: in double support, the *non support foot* is maintained motionless. During the single support phases, the foot is lifted and landed on the target footprint. While the shape of the motion is not important, we have to abide by the hypotheses of the simplified dynamics for the biped walk. These stipulate that the angular dynamic momenta of the bodies composing the robot are null. This condition cannot be satisfied, we focused on the smoothness of the trajectory and the speed of the motion. We opted for minimum-jerk motion[Shadmehr and Wise 2005] planning to lift and displace the non-support foot. Furthermore,the rotation of the foot while being moved was restrained to yaw around the ground normal. We found that planning footsteps with a single support phase of 0.8 to 1 second and a double support phase with 0.1 to 0.2 second produced very little difference between the planned ZMP and the achieved ZMP (calculated using 5.4).

With these tasks occupying first priority layer, the desired tasks follow starting from the second. We implemented Algorithm 2 together with the preview controller described in previous section to provide a comprehensive tool for whole body motion generation including dynamic stepping (see section 1.4 for further details). . In the following, we report a few examples of research works where this unified framework was used.

5.2 Experiments

Simultaneous manipulation and stepping

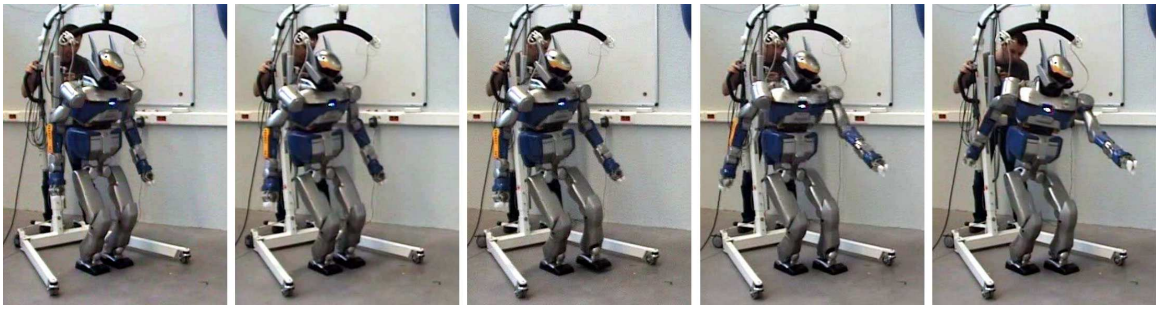


Figure 5.6:

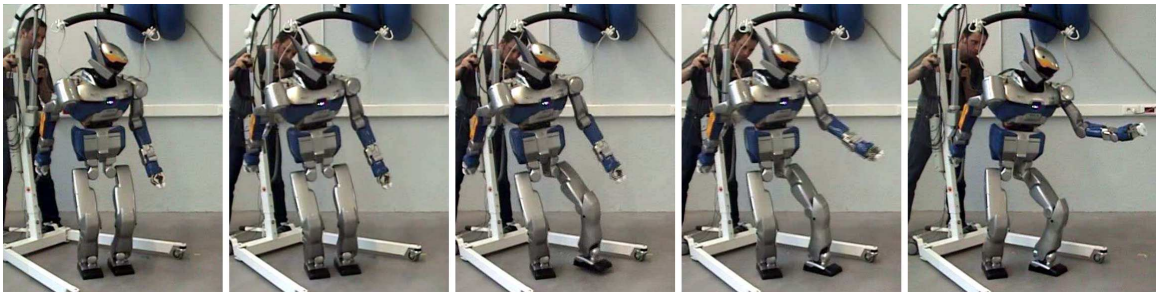


Figure 5.7:

This experiment illustrates the unified framework, first presented by [Yoshida et al. 2006]. The frames in figures 5.6 and 5.7 show two scenarios. In the first, a step and a whole-body motion is calculated to reach an object directly in front of the robot. In the second, the motion is for an object on the left side of the robot. In both cases a heuristical algorithm was used to derive the position and orientation of the step from the reach task. The motion was dynamically stable, despite the fact that an upper body motion is applied, while the simplified dynamic model required that the motion of the system of rigid bodies should be a translation on a fixed horizontal plane. To achieve this, the motion of the had was planned in workspace as a minimum-jerk trajectory preserving the continuity of velocity and acceleration.

Looking for and retrieving dictated target

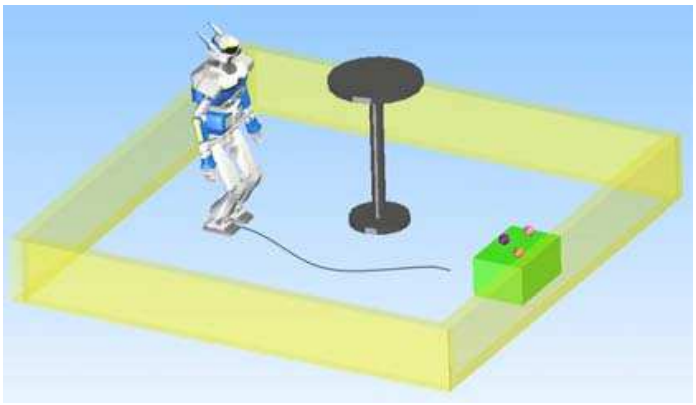


Figure 5.8: Overview of scenario: the robot must fetch one ball located on the green box at the other end of the room.

Figure 5.9: Frame 1 and 2: table detection. Left image shows the HSV image and right image is the back projection of the table color model in the source image. The rectangle locates the objects found. Frame 3 and 4: purple ball detection.

This experiment, presented by [Yoshida et al. 2007], was built around four components: voice recognition, perception of colored targets through stereoscopic vision, dynamic walk generation and whole-body motion planning including stepping. The user tells the robot to locate a box in the scene. When the robot reports successful recognition of the object (figure 5.9), it is told to go near it. The robot calculates and executes a dynamic walk motion using the preview controller presented in previous section. Near the box, the robot is told to locate a ball of a certain color. If successfully located, it is then told to pick it up and the robot calculate a whole-body motion to achieve the task. If necessary, a single step is added to the motion to make the task feasible (according to the heuristical solution presented by [Yoshida et al. 2006]). Figures 5.10 and 5.11 show a few frame samples from two instances of this experiment.

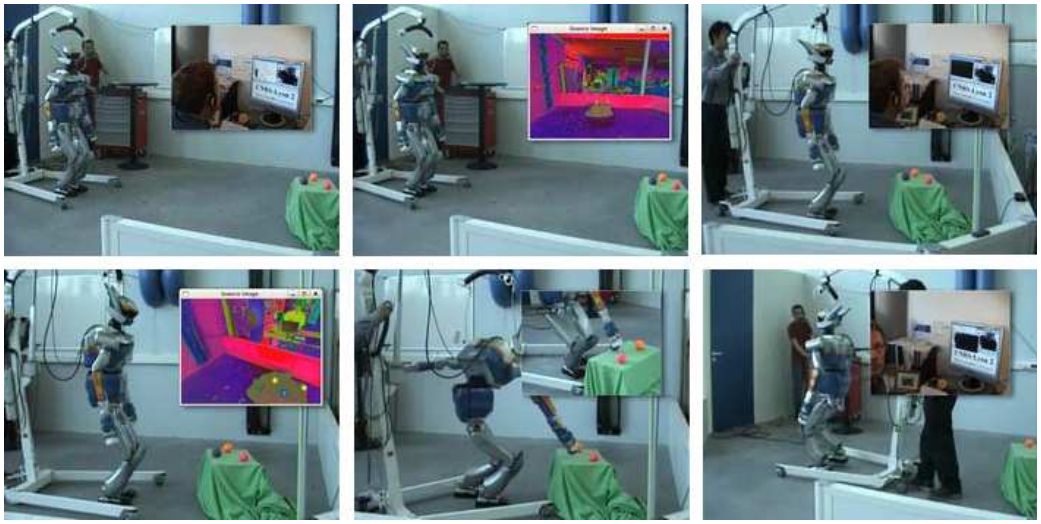


Figure 5.10: First experiment instance with remote control (the user was in remote city) of the robot through natural language interaction. The robot is asked to go to the table, grasp the ball there and come back to the initial position.

Figure 5.11: Second experiment instance. The robot is asked to give the purple ball to the person sitting on a chair. It walks toward the table, avoiding known obstacles in the environment. It first fails to grasp the ball (image 3 and 4), but with a simple error recovery strategy it finally succeeds (image 5) and accomplishes the mission.

Moving a bulky box

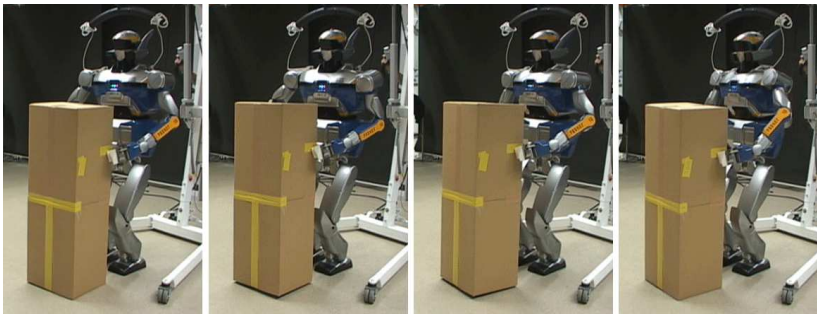


Figure 5.12: A sequence of three elementary rotations realized by whole-body motion in pivoting-based manipulation. The object is first inclined to stand on one corner (frame 2), then rotated along the vertical axis (frame 3) and finally inclined back to horizontal stance.

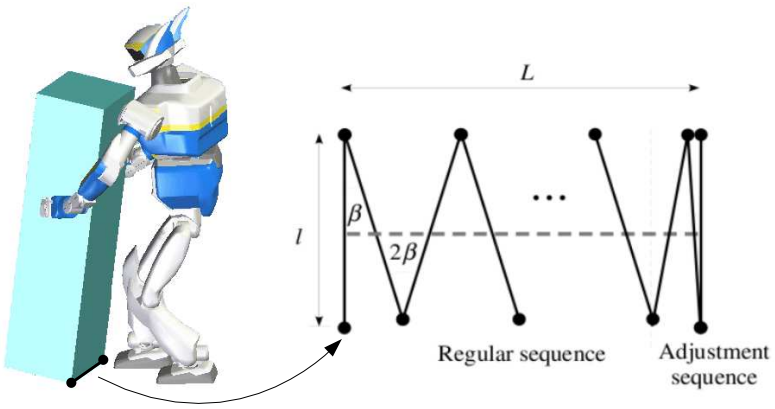


Figure 5.13: Transforming a straight line segment path into a pivoting sequence. The pivoting sequence is planned using rotation of the endpoints of the supporting edge.

This experiment by [Yoshida et al. 2008] emphasizes on the capacity of a humanoid robot to perform tasks that are difficult for other types of robots. It deals with manipulation of bulky objects. The task of the robot was to move a voluminous box across the workspace. The authors studied a strategy based on successive pivoting of the box around its corners and proved controllability properties ([Lamiraux and Laumond 2000; Yoshida et al. 2007]). These properties allow the authors to plan a successful motion of the box from start to goal position and orientation, then plan the position and orientation of both hands accordingly. These tasks are given to the inverse kinematics solver which computes whole body motion to realize the motion. The stepping is triggered as required: the authors monitor the position and orientation of the robot with respect to the box it manipulate and calculate a footstep when judged necessary. The footstep's position and orientation is customized such that the robot stays close enough

to the box, in comfortable manipulation range.

Joystick-steered head

[Sreenivasa et al. 2009] worked on a method for remote steering of a humanoid robot by controlling its head position and orientation. The motivation behind this approach comes from research in the field of human neuroscience: in human locomotion it has been found that the head plays a very important role in guiding and planning motion[Berthoz 2002]. The presented algorithm converts the controls of a 6D mouse to desired translation and rotation for the head of the humanoid robot. Several criteria are monitored to assert the feasibility of the user commands. In case of non feasibility, a footstep is calculated to make the robot move such that the controls input by the user become feasible. The accent was not put on the method of generation of the footstep, which as in above experiments, was customized for a specific task: positioning and orienting the head. An analysis has been conducted to determine when and where to step with respect to the desired values of the head task.

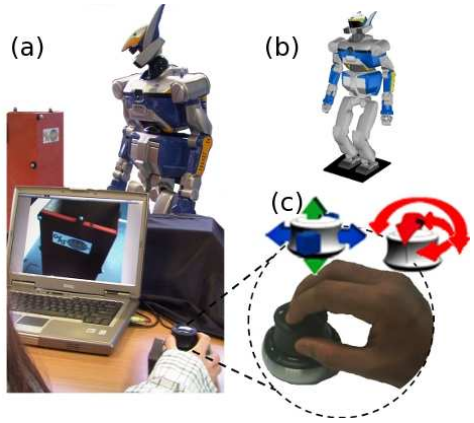


Figure 5.14: Extracted with author permission from [Sree09]. (a) Snapshot of user maneuvering HRP2 while viewing the output from its cameras. (b) The humanoid robot HRP2 in default position. (c) Magnified view of the 6D mouse with all available motion axes.

Figure 5.15: Extracted with author permission from [Sree09]. (a) HRP2 being maneuvered through space to find distant, hidden objects. (b) Plot of head position (solid line) and center of mass (CoM) position (dotted line) during motion. The zig-zag motion of the head and CoM position was due to the user alternatively looking left and right at every step.

6

Footsteps planning as an inverse kinematics problem

6.1 Principle of the approach

Consider a humanoid robot that made a few footsteps to stand where it could fulfill a task. Our approach is based on the following observation: a footstep can be parametrized by a translation and a rotation of the target footprint from the support footprint. As such, two successive footprints can be considered as two virtual bodies joined by a virtual link with three degrees of motion freedom. Connecting the footprints two by two, we obtain a virtual kinematic chain describing the walk path of the robot as a deformable articulated structure. By linking this chain to the robot at one of its feet, we establish a direct relationship between the tasks and the configuration of the walk path. We have a composite kinematic structure, on which certain constraints need to be observed and for which kinematic tasks are to be completed. All that remains is to apply the framework of numerical resolution reviewed in the first chapters.

The problem thus formulated becomes a problem of inverse kinematics. Any task that can be solved with inverse kinematics may be considered an input and prioritized in a stack. As for a robotic arm taking the shape that satisfies the desired position and orientation of its end effector, we expected the walk path derived by this method to be shaped directly according to the given tasks. We verified this property on a variety of scenarios, illustrated at the end of this chapter. Additionally, we had the idea to connect several instances of this virtual kinematic structure in series and optimize the walk path for tasks which are consecutive in time.

6.2 Construction of inverse kinematics problems

To define the problems of inverse kinematics associated with each case of footstep planning, we will define the subject kinematic structure, the set of permanent kinematic constraints and the set of prioritized tasks. We will make use of Algorithm 4 presented in section 3.4.3. We will denote a call to this algorithm by:

$$\delta q = solve(\{C\},\{T\},q) \tag{6.1}$$

for a kinematic structure of configuration $q \in \Re^n$ subject to the set of constraints $\{C\}$ and considering the set of prioritized tasks $\{T\}$.

6.2.1 A single footstep

After making a single step without slipping with its support foot, the humanoid robot has modified its support polygon by changing the position and orientation of its stepping foot on the ground. Our focus is not on the actual motion of the stepping, which require a shift of the body balance on the support foot, a displacement of the foot and a repositioning of the balance. We want first to known the configuration of the target support polygon that motivates the stepping. To do so, we associate a 3-degree-of-freedom configuration to the support polygon to denote the position and orientation of its moving footprint with respect to the support footprint. A 2D frame is associated with each footprint and the parameters of the configuration are denoted $(\Delta x, \Delta y, \Delta\theta)$ (see figure 6.1).

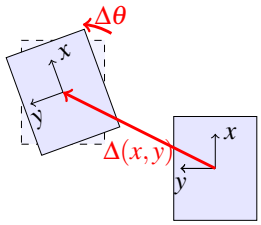


Figure 6.1: Degrees of freedom for one footstep

Before considering making a step, we have probably evaluated that the tasks to be done were not feasible otherwise. Recall the constraints that were applied to solve the scenarios of section 4.3 which supposed a fixed support polygon:

- Joint limits
- Static equilibrium
- Self-collision and obstacle avoidance
- Fixed position and orientation of feet

Presently, we consider the robot in what we may call a *planning mode* where we try to solve the given set of tasks without modeling the friction on the ground. Then, instead of imposing null linear and angular velocities on the stepping foot, we authorize a translation along the plane of the ground and a

yaw rotation around its normal $\vec{z}$. By doing so, we relax the last constraint of the above list and allow the support polygon to accommodate to the motion of the legs moved by the tasks. At this time, we have not yet augmented the state of the kinematic structure of the robot. We have simply allowed a virtual motion of a foot on the ground. However, there are additional constraints to be observed due to this relaxation.

First, the feet must not collide during the planning motion. This may be taken into account at the level of self-collision avoidance constraints if they cover all pairs of bodies in the robot, or by setting ad-hoc constraints on the feet. In our case, we chose the second solution consisting in forbidding a foot to enter a half space containing the other foot (see figure 6.2). This constraint is achieved by damping the velocity of the upper and lower interior corners of each foot near the border of other foot's half space. The calculation of the corresponding linear systems follows (4.19) and (4.20).

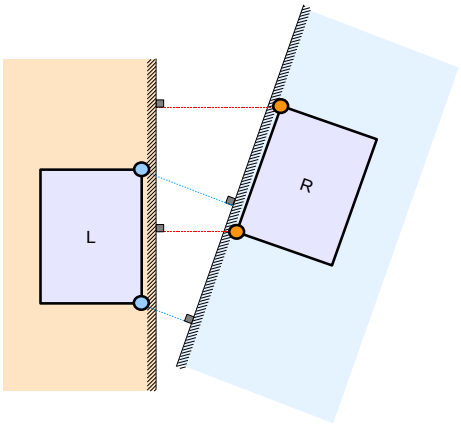


Figure 6.2: Each footprint lies in an associated half space where the other foot cannot enter. This constraint is achieved by damping the velocity of the upper and lower interior corners of each foot near the border of other foot's half space.

Second, the calculation of the linear system for the static equilibrium constraint changes since the support polygon is now deformable. Recall the static equilibrium constraint:

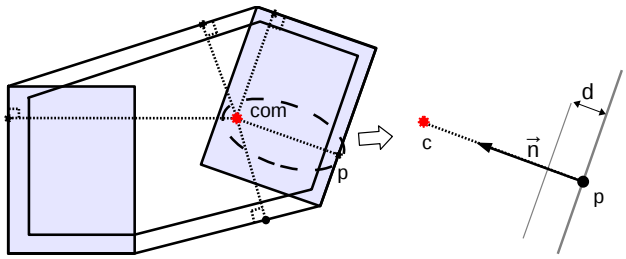


Figure 6.3: The velocity of approach of the center of mass projection to the edges of the support polygon is damped.

$$-\langle \delta(\overrightarrow{PC}) | \vec{n} \rangle \delta q \leq \lambda (\langle \overrightarrow{PC} | \vec{n} \rangle - d)$$

(see illustration 6.3). The point P, projection of the center of mass on an edge of the support polygon, is no longer necessarily fixed, therefore the differentiation of the constraint leads to:

$$\langle J_P - J_C | \vec{n} \rangle \delta q \leq \lambda (\langle \vec{PC} | \vec{n} \rangle - d)$$

where J_P is the jacobian of the position of point P. In case P belongs to the edge of a support foot, the jacobian J_P is null and the linear system is the same as the one calculated in (4.20).

Third, the support polygon must be feasible at all times in quasi-static motion. To achieve this, an admissible stepping region is defined for each foot with respect to the other. We have, as a matter of fact, to constrain both feet to keep the support polygon valid for further stepping with any foot. The shape of the admissible stepping regions depends on the robot's capabilities. Since we generalize this method to any number of steps, we did not judge useful to accurately estimate the maximal stepping region of our robot and we set rather restrictive bounds. If one small step is not enough to solve the task, we increment the number of steps (more on this matter will follow in the next section). Consider the region defined by the simple bounds:

$$\Delta x_{min} \leq \Delta x \leq \Delta x_{max}$$
$$\Delta y_{min} \leq \Delta y \leq \Delta y_{max}$$
$$\Delta \theta_{min} \leq \Delta \theta \leq \Delta \theta_{max}$$

Figure 6.4 shows the authorized area that results from the bounds. The symmetry of the bounds is applied to left foot with respect to right. The calculation of the linear inequality systems follow (4.3) and (4.4).

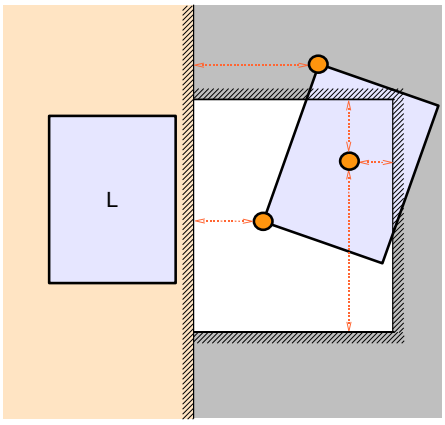


Figure 6.4: Stepping region for the right foot. The gray region is forbidden to the center of the right foot. Anti-collision constraints bound the stepping region from the left. The constraint on yaw rotation is not represented here.

Let us summarize the set of constraints $\{C\}$ used in this case:

- Joints limits
- Static equilibrium (with deformable support polygon)
- Support polygon configuration bounding
- Collision avoidance (including feet)
- Feet sliding on the ground (only horizontal translation and yaw rotation allowed)

Given a stack prioritized tasks, we iterate the resolution of (6.1) until convergence. If the final robot configuration does not achieve its primary task, we may want to consider re-planning with more than one footstep.

6.2.2 Several footsteps

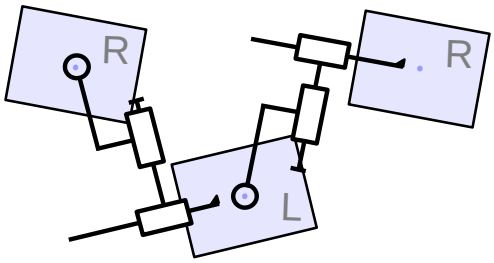


Figure 6.5: Virtual kinematic chain linking successive footsteps

We consider the general case where the robot makes $p > 1$ steps to fulfill its tasks. We view two successive footsteps as two virtual links joined by a joint possessing three degrees of motion freedom (figure 6.5). Define:

$$Q_i = (\Delta x_i, \Delta y_i, \Delta \theta_i) \tag{6.2}$$

the configuration of the support polygon of the i-th step. We define the augmented robot state

$$\tilde{q} = (Q_1, ..., Q_{p-1}, q) \tag{6.3}$$

where we remind that q is the vector of configuration of the robot. $\tilde{q}$ is the configuration vector of the kinematic model of the robot augmented with a virtual kinematic chain of footprints (represented in figure 6.6).

Considering the permanent set of constraints for the inverse kinematics problem, those are comprised of:

- List of constraints seen for the single step case.
- For each additional step:
  - Support polygon configuration bounding
  - Collision avoidance between footprints and with environment.

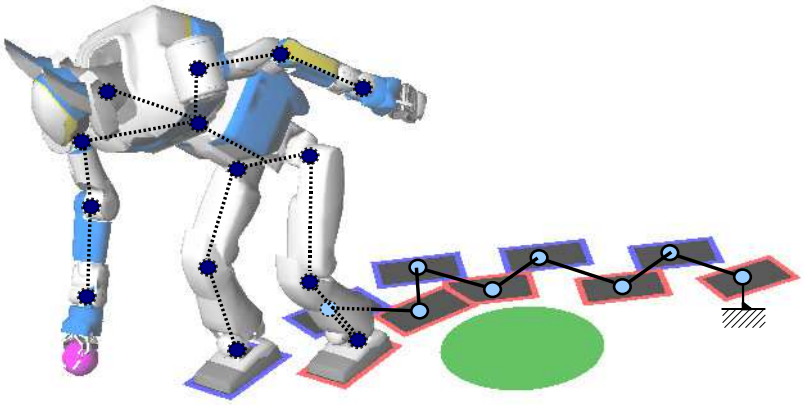


Figure 6.6: The kinematic model of the robot is augmented with a virtual kinematic chain of footprints.

– Footprints sliding on the ground

The difference in this case lies in the fact that the jacobians involved in the linear systems derived from constraints and tasks are calculated with respect to the entire kinematic structure. The same algorithm 6 applies. Call $\{\tilde{C}\}$ the set of constraints and $\{\tilde{T}\}$ the set of prioritized tasks. By repeating the operation:

$$\delta\tilde{q} = solve(\{\tilde{C}\},\{\tilde{T}\},\tilde{q}) \tag{6.4}$$

we are solving a prioritized inverse kinematics problem numerically, gradually unfolding our composite kinematic structure from the start support polygon to the configuration allowing the robot to achieve the tasks.

Let us now notice that we have supposed the number of steps p known. In the general case, this information is not accessible, even though it can be estimated. We have also assumed that the first stepping foot has been chosen before starting the numerical resolution. These parameters can be chosen objectively, as described in the following section.

6.3 Tuning the parameters

6.3.1 Number of footsteps

At a given iteration of the inverse kinematics problem designed for several footsteps, the kinematic structure is fixed and so is the number of footsteps. This does not prevent us from modifying the kinematic model for the following iterations. There are several factors that can be considered to decide on augmenting or diminishing the virtual chain of footprints. A first method is summarized by Algorithm 7 and illustrated in figure 6.7: given a task $T(q) = 0$ at the top priority level, we start from a single step and add one whenever the decrease in the norm of task becomes too slow, i.e violating a condition such

as the following:

$$\frac{\|T(q(t+\Delta t)) - T(q(t))\|}{\Delta t} < min_{threshold}$$

where the function $T(q(t))$ represents the value of the task $T(q)$ at time t. The slowing of the convergence announces a local minimum for the inverse kinematics we are solving. Thus, it is natural to try to extend the accessible space of the kinematic structure by giving more "slack" to the virtual chain of footprints. This, however, results in a "tense" chain, that is to say, a walk path that is always stretched towards the support polygon required by the tasks.

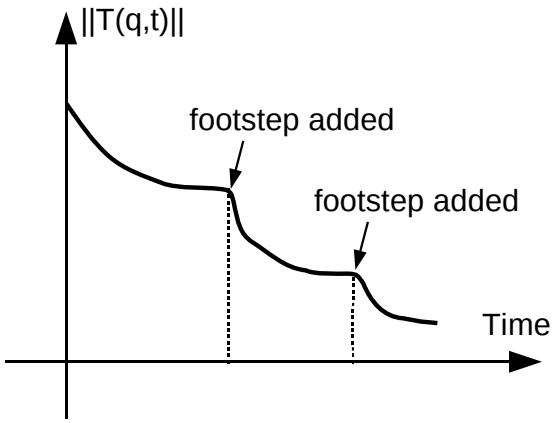


Figure 6.7: Adding footprints according to task progression.

Suppose that we solved a problem using this first method of adding footsteps as needed by the task. In a second phase, we seek to modify the shape of the path so that it has for instance, more steps, smaller treads, smoother curvature, etc. All there is to do then is to express these desired properties into a task and derive the corresponding linear system which is added in last priority, below all the tasks of the original problem. For instance, consider an ideal configuration $\tilde{q}_{ideal}$ for the kinematic structure and define the following posture task:

$$\tilde{q} = \tilde{q}_{ideal}$$

$\tilde{q}$ being defined by (6.3). Our aim is to reduce $\|\tilde{q} - \tilde{q}_{ideal}\|$ as much as possible while keeping the tasks solved. In order to do so, we start from the solution to the first problem, add this task at the end of the stack, then add a footstep, iterate on the new inverse kinematics problem until a local minimum is reached. We keep this process until addition of footsteps does not reduce the residual norm anymore. This is the same as Algorithm 7. The evolution of $\|\tilde{q} - \tilde{q}_{ideal}\|$ follows the example shown in figure 6.7.

Notice that it is possible to perform the resolution of the original problem and the optimization of the path's shape simultaneously. This can be useful for instance to avoid stretching the robot's structure before it arrives to effective action range.

The same idea could be applied to shorten the chain of footprints. Imagine that the resolution of

Algorithm 7 Solve tasks $\{T_1(\tilde{q}) = 0, ..., T_k(\tilde{q}) = 0\}$ and expand number of footprints as needed

Define ε_p the minimum task value progression required to continue
Call p the measure of task value progression
repeat

  Do **Block** 1**:**
  {
  for i from 1 to k **do**
    Calculate value of task i: $T_i^0(\tilde{q})$
  end for
  Solve prioritized linear systems (Algorithm 2)
  $\tilde{q} \leftarrow \tilde{q} + \delta\tilde{q}$
  for i from 1 to k **do**
    Calculate value of task i: $T_i^1(q)$
    Evaluate progression $p = T_i^1(\tilde{q}) - T_i^0(\tilde{q})$
    if $p > \varepsilon_p$ **then**
      break.
    end if
  end for
  }

  if $p < \varepsilon_p$ **then**
    Expand $\tilde{q}$ with one footprint: $\tilde{q} \leftarrow (Q_{new}, \tilde{q})$
    Redo **Block** 1
  end if

until $p < \varepsilon_p$

the initial problem started with a number of footsteps that turned out larger than necessary. To reduce this number, a posture task is applied to fold a number of undesired footprints back to the initial support polygon. Figure 6.8) illustrates this process.

6.3.2 Starting footstep

To get rid of the last a-priori information concerning which foot to start stepping with, we can make calculations for both available choices, left and right. If the tasks are solved for both cases, we may pick the choice that gives the least number of required steps. To avoid an arbitrary decision in case the number of steps is identical for both choices, we may select the start foot based on the result of an extra task that minimizes the deformation of the solution posture with respect to a reference *rest posture*. This task which writes as $q_{robot} = q_{rest}$ is systematically added with lowest priority in the stack of tasks. We keep the solution producing the smallest residual task value $\|q_{robot} - q_{rest}\|$. More sophisticated criteria may naturally be used instead of this criterion intended as a default one.

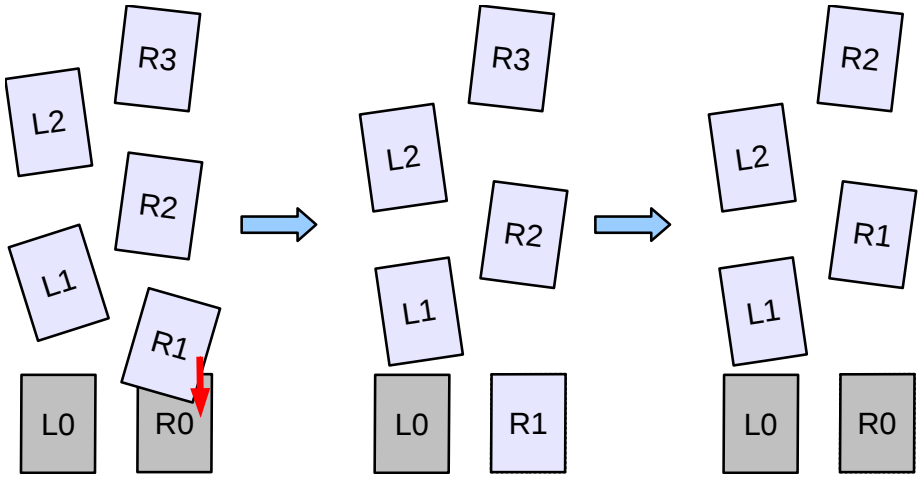


Figure 6.8: The footprint of the first step must be folded back to its initial position before it can be removed.

6.4 Illustration

We have applied our footstep planner to a variety of scenarios. The feet's relative position and orientation bounds were chosen as follows (left foot w.r.t right foot, the opposite case is taken symmetrical):

$$\begin{aligned} -0.22cm < \Delta x < 0.22cm \\ 0.07cm < \Delta y < 0.25cm \\ -0.1rad < \Delta\theta < \frac{\pi}{4} \end{aligned}$$

These bounds are small enough to guarantee quasi-statical stepping for the robot HRP-2. It is not necessary to estimate the maximal bounds unless one wants to minimize the number of steps to achieve a given task.

Reaching an object

In this scenario, the robot stands 2m from the target ball. A simple obstacle is modeled with a disc region and the feet are constrained to avoid the corresponding area placed on the ground. For this we use velocity dampers between the center of the obstacle and its projections on each footprint. Figure 6.9 shows the state of the kinematic structure at various intermediary steps from the iterated inverse kinematics problem. Initially, the virtual chain is folded down and all support polygons coincide with the start polygon. The chain unfolds continuously until the robot has either satisfied its tasks or the structure has been trapped in a local minimum. The second case may occur when the tasks are absolutely unfeasible, which points to a task planning problem or a physical limitation of the robot. It can also occur when the chosen number of steps is insufficient. This can be investigated with one of the methods

described above. For this test scenario, nine steps were needed to accomplish the task. This resulted in (28 degrees of freedom of HRP-2) + (3x9 degrees of freedom from the virtual chain of support polygons) = a total dimension of 55 for the state parameters.

Picking up from the floor

For this second scenario, the robot has to pick up an object that lies on the ground between its feet. To avoid generating a footprint on the target object, a virtual obstacle for the feet only is placed around it. Figure 6.10 shows a progression to the solution for this scenario. We planned an actual motion based on the foot placements to validate the method on the humanoid robot HRP-2. We used the method described by [Kajita et al. 2003] to calculate a dynamic walk motion.

Recovering a comfortable posture

In this third scenario, the robot has already performed a motion to look at the pink ball without making steps. The achieved posture is awkward and continuing to stare at the target in that shape is not very well looking. The robot may recover a comfortable posture by making a few steps. This scenario was presented in [Sreenivasa et al. 2009] with a heuristical method to derive the position of the required footsteps. We tackled the same problem in a generic way using our planner: we defined a comfortable posture q_{rest} for the robot and the posture recovery task written as:

$$q - q_{rest} = 0$$

q the configuration of the robot alone. This task was applied under the constraint of keeping to look at the ball. We presented the expression of this task in section 4.2. We found that four steps were needed to achieve a posture close to the initial configuration (see figure 6.12).

Consecutive tasks

In this final scenario, we create a more complex kinematic tree that is composed of alternations of virtual kinematic chains and robots. If the former design may be labeled *the centipede robot* this design might remind of the mythical creature Cerberus. Unlike it, however, this design allows complete separation of several robot instances with an arbitrary number of virtual footsteps in between. The aim of this design is to plan a single walk for a succession of tasks. The configuration of the kinematic structure resembles:

$$\tilde{q} \quad = \quad (q_{walk1}, q_{robot1}, ..., q_{walkN}, q_{robotN})$$

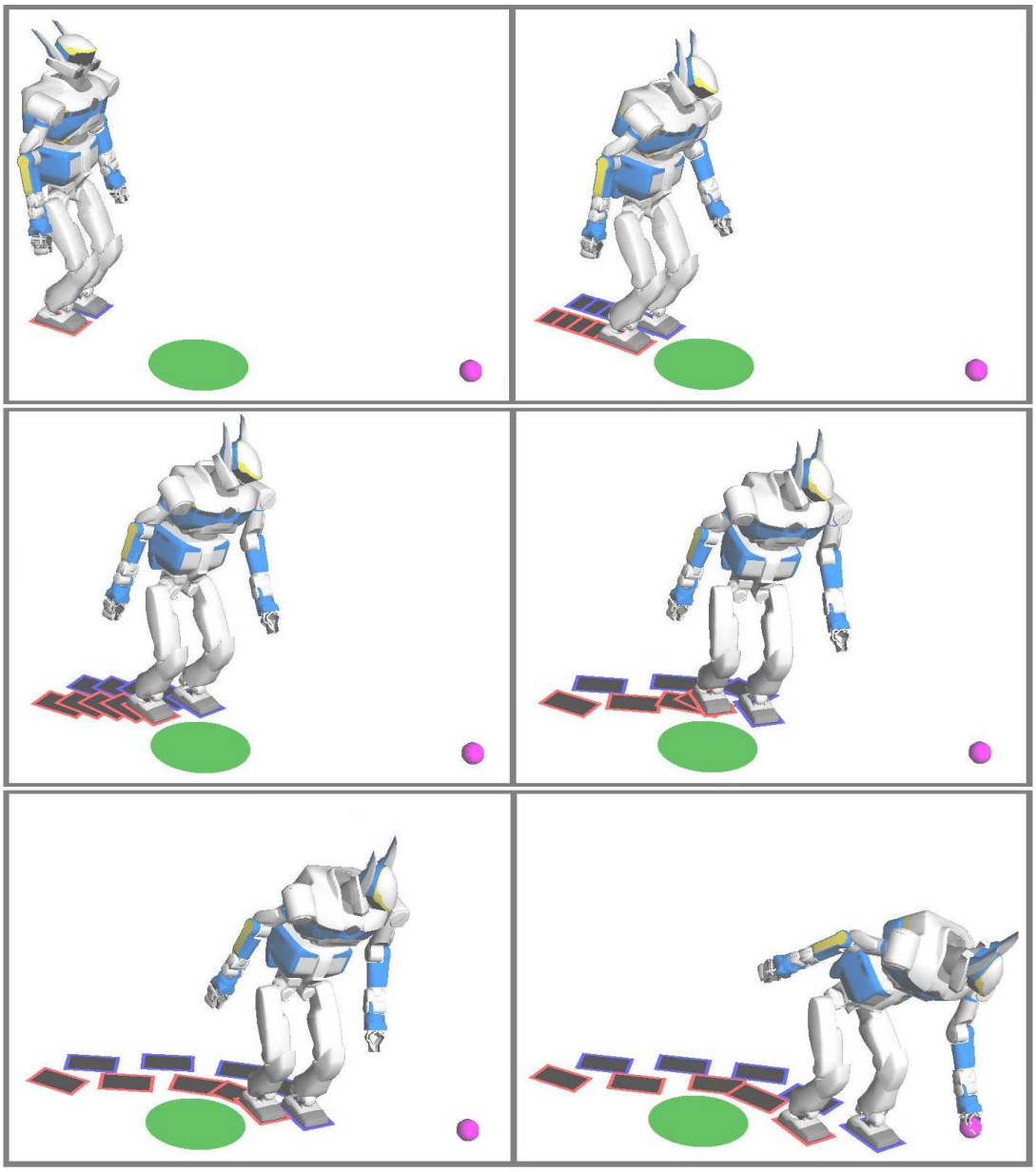


Figure 6.9: States of the full kinematic structure at different steps of the optimization. The task is to grasp the pink ball without stepping on the green region (obstacle). The last view shows the solution footprints retained for the actual robot locomotion.

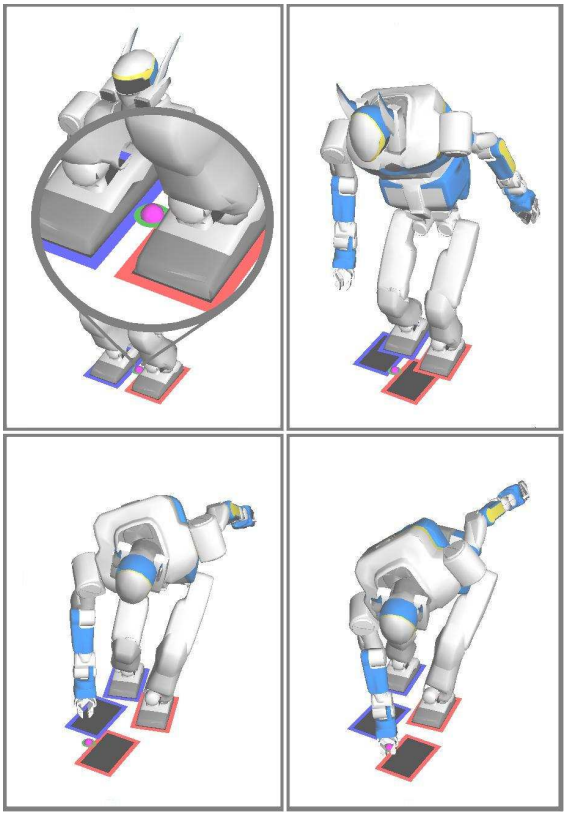


Figure 6.10: Planning footsteps to pick up an object on the floor. A virtual obstacle (green disc) is added around the object to avoid stepping on it.

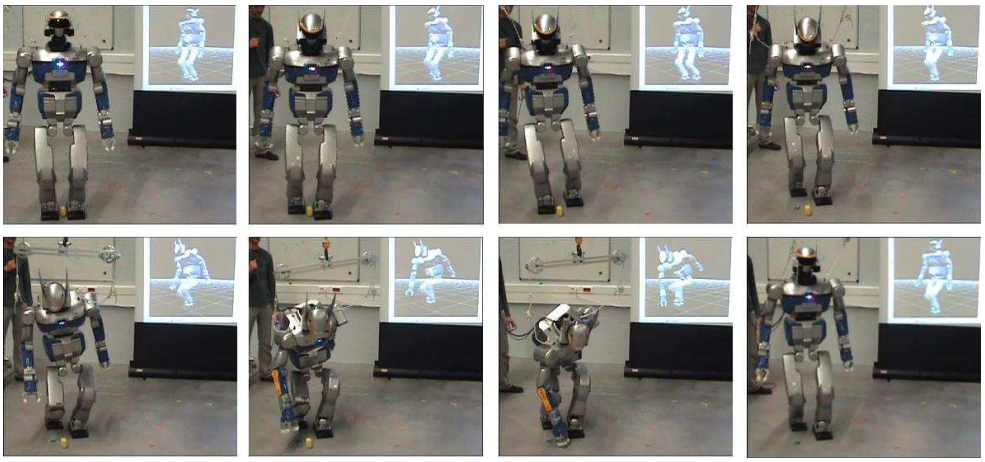


Figure 6.11: HRP-2 picking up an object (in yellow) lying between its feet. First a dynamic walk is planned over the support plygons produced by our local foot placement planner, then the whole body is driven by a reaching task under permanent self-collision avoidance constraints.

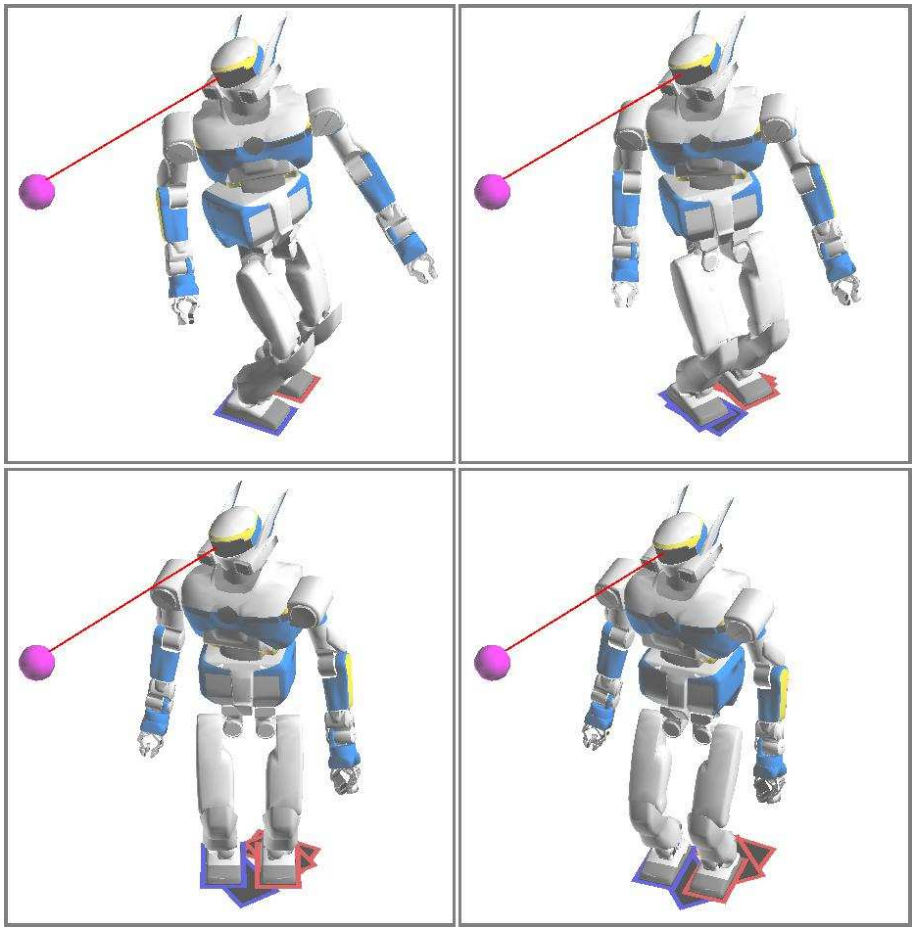


Figure 6.12: Planning footsteps to recover a comfortable posture in the middle of another task (here to look at the ball).

And the tasks to achieve are specified for each robot:

$$\begin{aligned} T_1(q_{walk1}, q_{robot1}) &= 0 \\ T_2(q_{walk1}, q_{robot1}, q_{walk2}, q_{robot2}) &= 0 \\ &\dots \end{aligned}$$

The scenario on which this idea was tried was the following: the robot is facing a trash bin which is 2.5m from him, there are two objects to be collected on the floor and thrown in the bin, one slightly to the right, the other to the left. The kinematic chain was built as a sequence of 4 footsteps, a first robot, 2 footsteps, a second robot, 2 footsteps and the final third robot. The tasks were three: robot1 collects the right object with the right hand, robot2 collects the left object with left hand and third robot places both of his hands over the trash bin. Starting from the initial position where all robots and footprints were superimposed over the start support polygon, we specify an inverse kinematics problem as in the previous scenarios to solve the set of tasks. The result is shown in the last two frames of figure 6.13.

As expected the computation time was much higher than in previous cases due to the very large number of parameters involved. Yet, as the illustration shows, the final walk path is optimized for the sequence of tasks. The path is little deviated by the first and second collections and is rather straightly oriented to the bin.

If the computation time is an issue, the problem can be tackled differently. The tasks are solved separately one after the other, first the right collection, second the left collection starting from first task's support polygon, and last the disposal task starting from second task's support polygon. Thus, the involved number of parameters is roughly divided by three and a solution for each task can be found quickly. The final phase will then follow to connect the successive solutions together, thus building the complex kinematic structure described above, then solve the posture task described by (6.3) to optimize the number of footsteps. The advantage of this algorithm lies in the fact that the heavy optimization aspect is left to the end: once we have a basic sub-optimal solution answering the three tasks, we iterate on the optimization of the full path until convergence, or until the time we allocate for this extra optimization is over, then the last reached state is picked as the final path.

6.5 Conclusion

We have presented an inverse kinematics formulation of the footsteps planning problem. The tasks are directly related to the shape of the walk path and the final support polygon guarantees the fulfillment of the tasks.

As expected, the computation times grew rapidly as the footstep planning problems required higher number of parameters. Our planner may not be suitable for tasks requiring a long locomotion. For such tasks, one would prefer an algorithm that plans a walk path to a remote goal position and orientation[Yoshida et al. 2008] or that allowa a human operator to interviene to guide the robot near close to where it could perform its tasks [Chestnutt and Kuffner 2004]. Our method is best used as a fine-tuning algorithm that takes over the end of the planned walk path and reshapes the last few steps

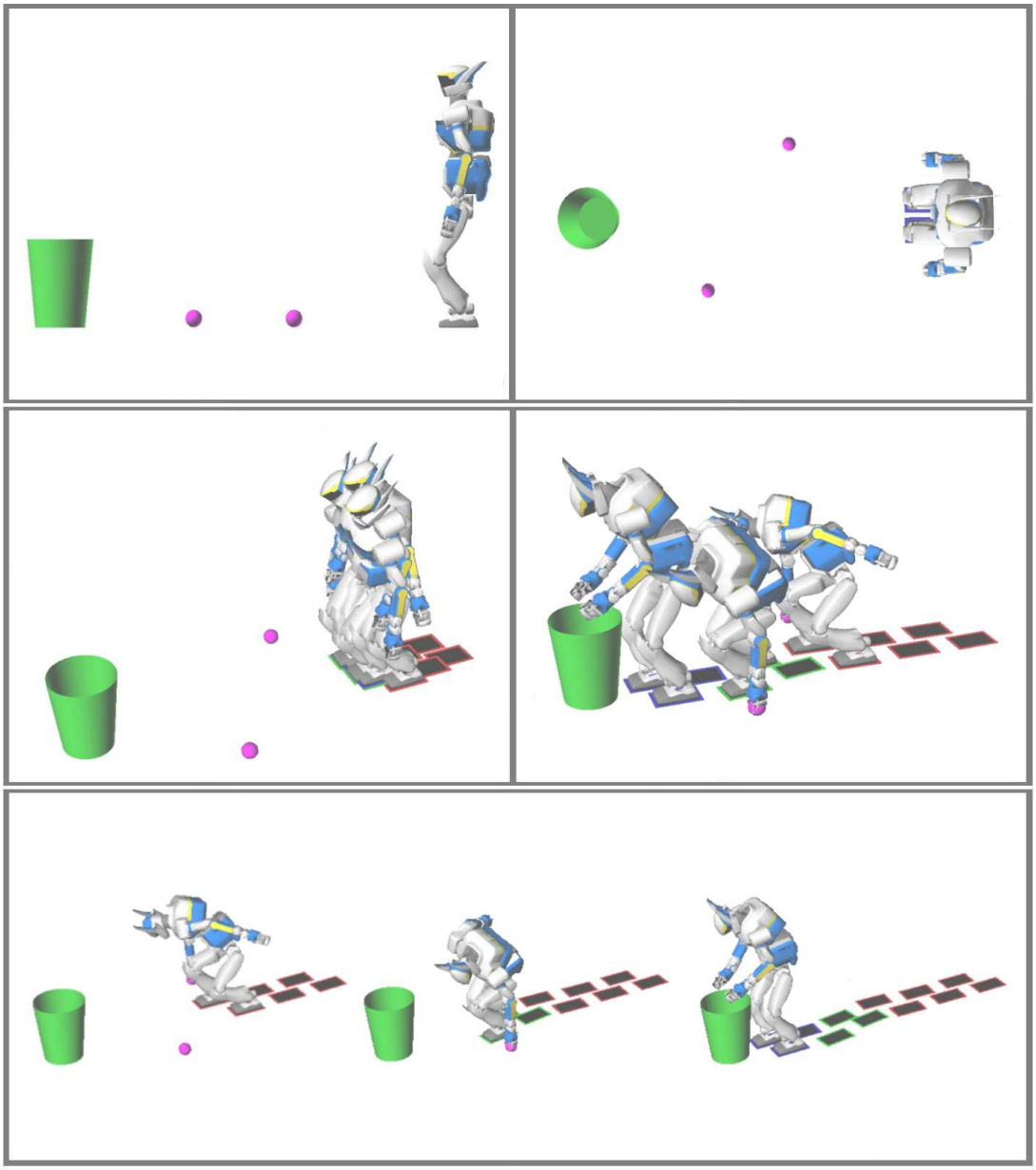


Figure 6.13: Planning footsteps for three consecutive tasks: pick right then left ball from the floor and put them in the bin. Three virtual kinematics chains of footprints and robots are connected in series (colored in red, green, and blue), each being affected with a task. The solution path is optimized, there is little zig-zag from the right to the left ball and then to the bin.

precisely according to the tasks.

Despite the fact that the footsteps are planned on a flat terrain, we believe we have advanced a robust method to relate whole body motion to locomotion.

7 Integration

7.1 The full algorithm

Let us summarize the elements at our disposal:

- A wide palette of kinematic tasks
- A generic algorithm that solves a set of tasks with priority
- An generic algorithm which plans the footprints based on a set of tasks
- A method to calculate dynamic or quasi-static walk motions for series of footprints.

We made a few connections between these elements, described in figure 7.1, to build an algorithm that computes a whole-body motion and locomotion which are task-only dependent.

The algorithm starts by trying to solve the given set of tasks from the initial robot configuration q_0, using Algorithm 5. If it fails, the last configuration reached be the robot is augmented with virtual footstep link and given to Algorithm 7 which, by solving the tasks for the virtual structure, plans the needed footsteps for the tasks. At this point, the algorithm might fail solving the requested tasks (End 2 in the figure). The reasons have been reviewed in previous section, they could relate to a conflicting tasks definition (reaching with both hands two objects that are two far with the same priority), inadequate geometry of the robot (reaching an object placed too high), singularity of a task with respect to the configuration (an arm is all stretched out while a task is requested to position the hand on the shoulder). We would need a higher level component to assess the feasibility of tasks and potentially plan an intermediate set of before coming back to solve the original tasks.

If, on the contrary, the footsteps planner succeeds in producing a walk path which fulfills the tasks, the result is given to the locomotion planner, seen in section 5.1.4. The latter calculates stepping motion

in quasi-static or dynamic mode which moves the robot to the last support polygon, adapted for the tasks. Finally, a satisfying whole body motion is generated using Algorithm 5.

The advantage of this algorithm for the developer is obvious: it deals with the redundancy of the system, observes the priorities between the tasks and takes into accounts the locomotion ability of the system to help achieve a task. The developer can thus focus on higher level planning, (or higher level of artificial intelligence as we called it in the introduction) without having to design specific strategies to coordinates locomotion and manipulation. We describe in the following how we used this algorithm to solve a full scenario.

7.2 Illustration

We sketch hereby a realizable field experiment. In the scenario, the robot must take cups from shelves and place them on a table. Successive sub-tasks must be planned to fulfill this objective.

The first task is to grasp the cups. The reason why the robot sees the targets is that there is a direct space volume free of obstacles between the cameras and the cups. We observed that if we kept a little free space around its body, the robot was always able to place its hands in its vision field . Therefore, a strategy to reach for any seen object could be the composition of two motions: a first motion to make the hand enter the direct volume between the cameras and the targets, then the reaching motion under collision avoidance constraints.

However, it will be very difficult for perception algorithms to accurately model the shelves from the initial position of the robot, two meters from the targets. We try first to bring the robot closer to the cups. The initial problem is thus a simplified one where the footprints are planned such that the hands reach for the cups without taking the arm-shelf collisions into account (see figure 7.3). The planned footprints are constrained not to collide with a cylinder around the table and a vertical plane in front of the shelves. The footprints are further more enlarged such that the real locomotion of the robot, involving a sway from a foot to the other, does not produce body collisions with surrounding objects.

The robot is now in front of the cups, the free vision volumes separating the cameras from the cups can be better identified, a whole-body motion is calculated such that the position of the hands come in inside the volume (figure 7.4). The second part of the strategy presented above consists then in calculating a whole-body motion to reach for the cups while avoiding collision. The orientation of a hand is controlled with three tasks: two tasks of type *coplanar* (4.13) to ensure that the axis of the cup becomes in the sagittal plane of the cup, a third task of type *tilt* to define a solid angle containing the possible directions of the vertical axis of the hand (4.11)). The resulting motion is shown in figures 7.4 and 7.5.

To draw the cups back, the robot has to pull the hands out of the shelves. However, to avoid dragging the cups on the shelf, they are wrapped in a simplified geometry (cylinder capped with half spheres) that purposefully intersects the shelf. A collision avoidance task (not constraint) is set in first priority such that, in absence of any other task, the cup is slightly pushed above the shelf. Equality position tasks are placed in second priority to draw the cups out while they are being pushed by the anti-collision task

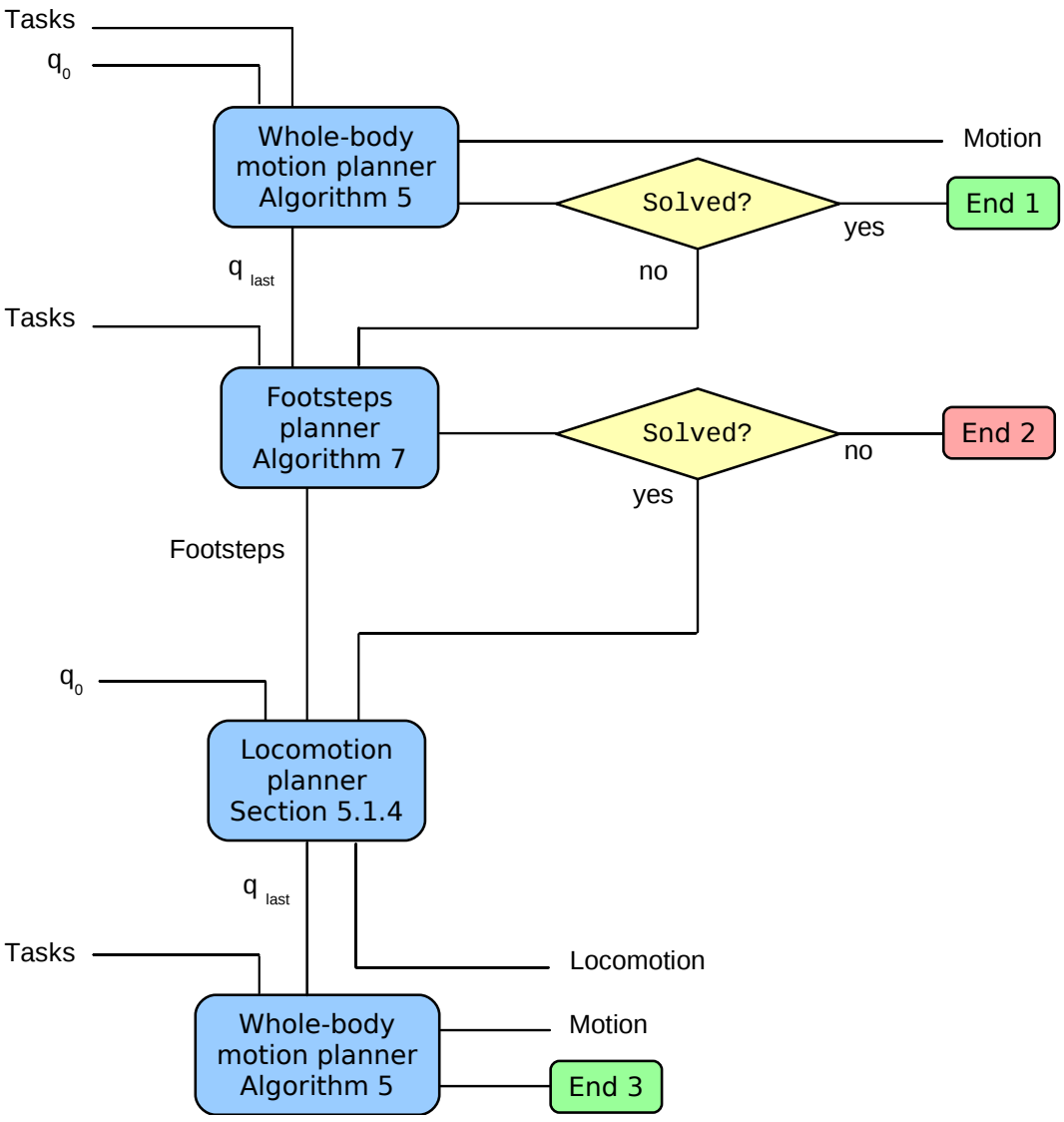


Figure 7.1: The full algorithm for task-driven motion planning

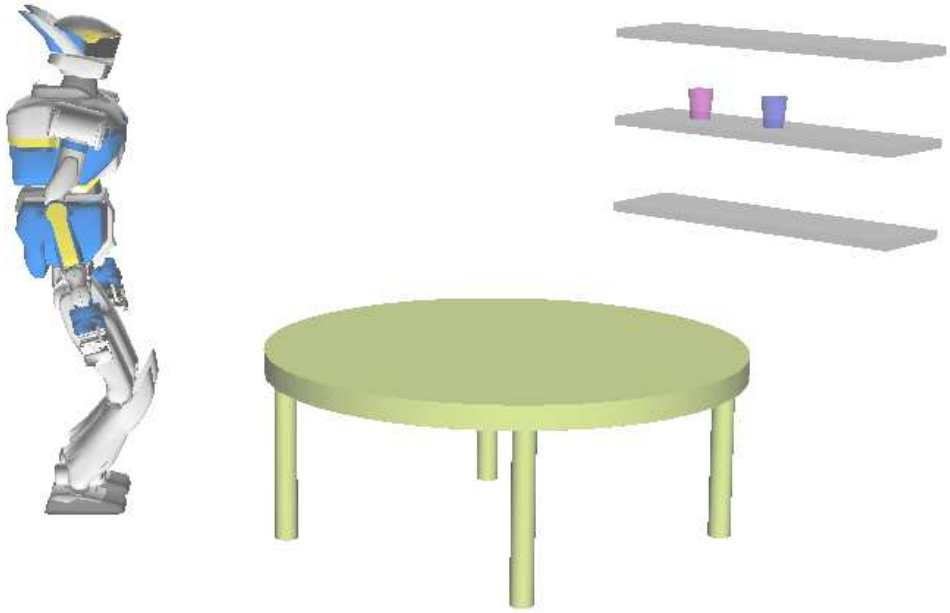


Figure 7.2: Field scenario: the robot has to place cups on the table.

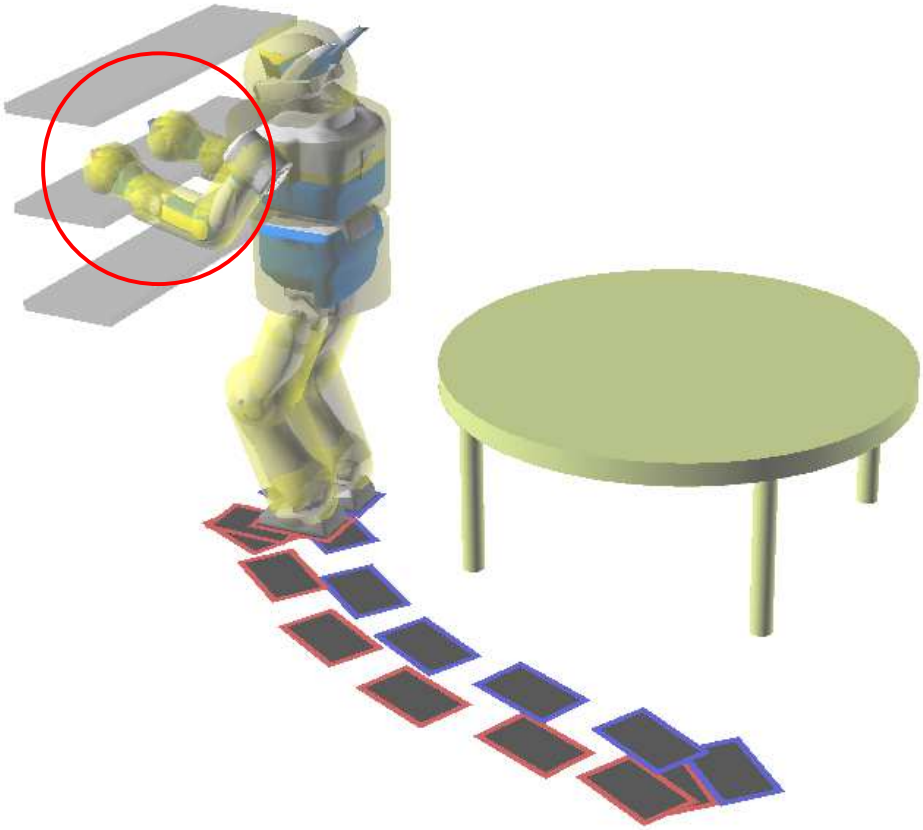


Figure 7.3: Phase 1: plan the footsteps that bring the robot close to the cups. The footsteps must not make the robot collide with the table and the shelves, which can be enforced through constraints on the footprints alone.

(figures 7.6 and 7.7).

The remaining task to fulfill the objective is to turn around and leave the cups on the table. In this particular situation, the algorithm did not find a solution for the task of looking at the table (necessary to identify a free manipulation volume). The robot is stuck (figures 7.8) in one of the local minima to which most numerical algorithms are prone. The reason here is simple: the space between the front plane of the shelves and the table is actually very narrow for an on-spot turn. We have to resort to an additional algorithm to avoid this limitation of the footstep planner. One can for instance use a random two dimensional shooter and quickly end up in the situation of figure 7.9.

From the new position, the robot sees the table, identifies a useful volume for its tasks and is able to place the cups (figure 7.10). We have reached this result through usage of our algorithm coupled with a

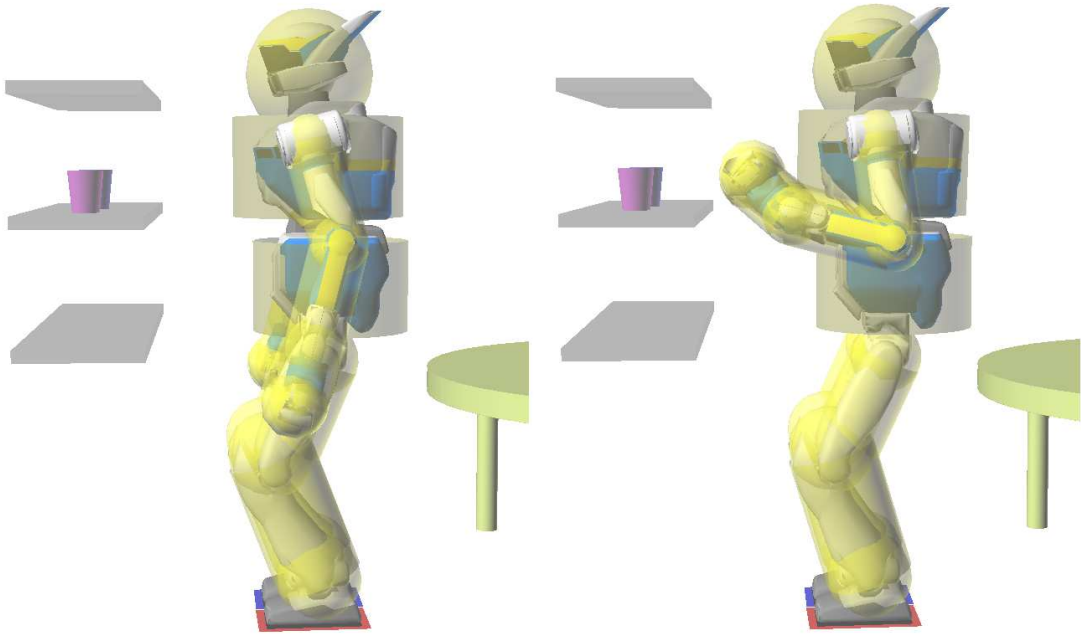


Figure 7.4: Phase 2: place the hands in the vision field, where a an obstacle-free volume is implicitly identified between the cameras and the targets. Collision with self and obstacles is avoided by wrapping the pairs of potentially colliding entities in simplified geometry.

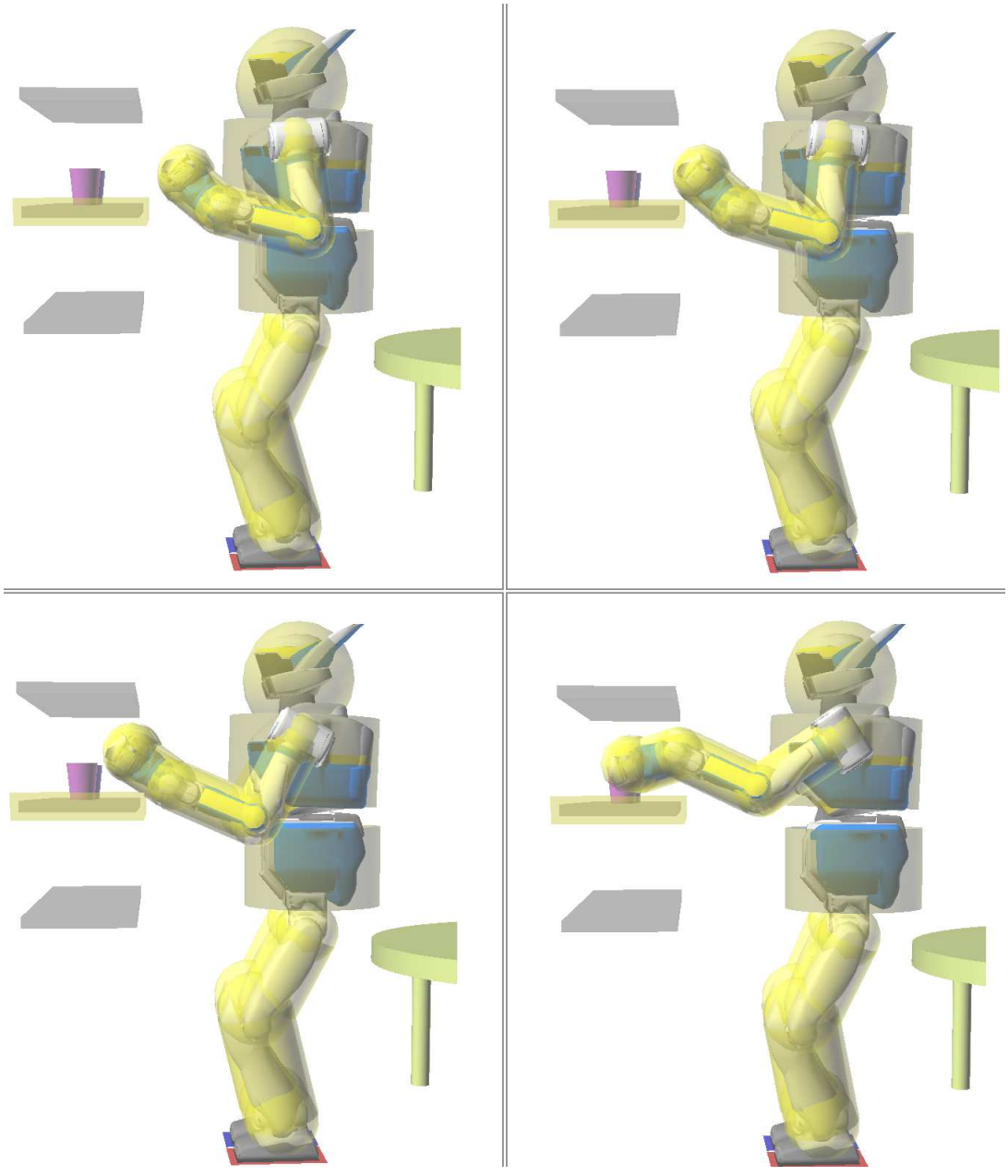


Figure 7.5: Phase 3: reach for the targets while avoiding avoiding obstacles.

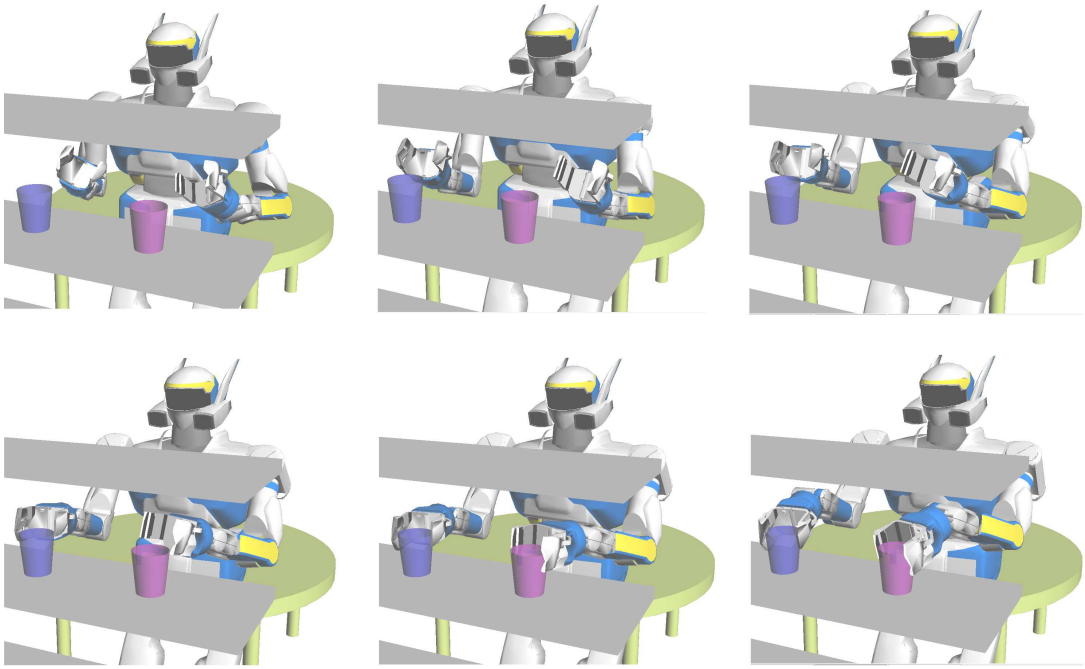


Figure 7.6: The final orientation of the hands with respect to the cups is not specified: tasks are set such that a hand's grasping plane contains the axis of a cup.

very simple random shooter that could be interpreted as a higher-layer of intelligence.

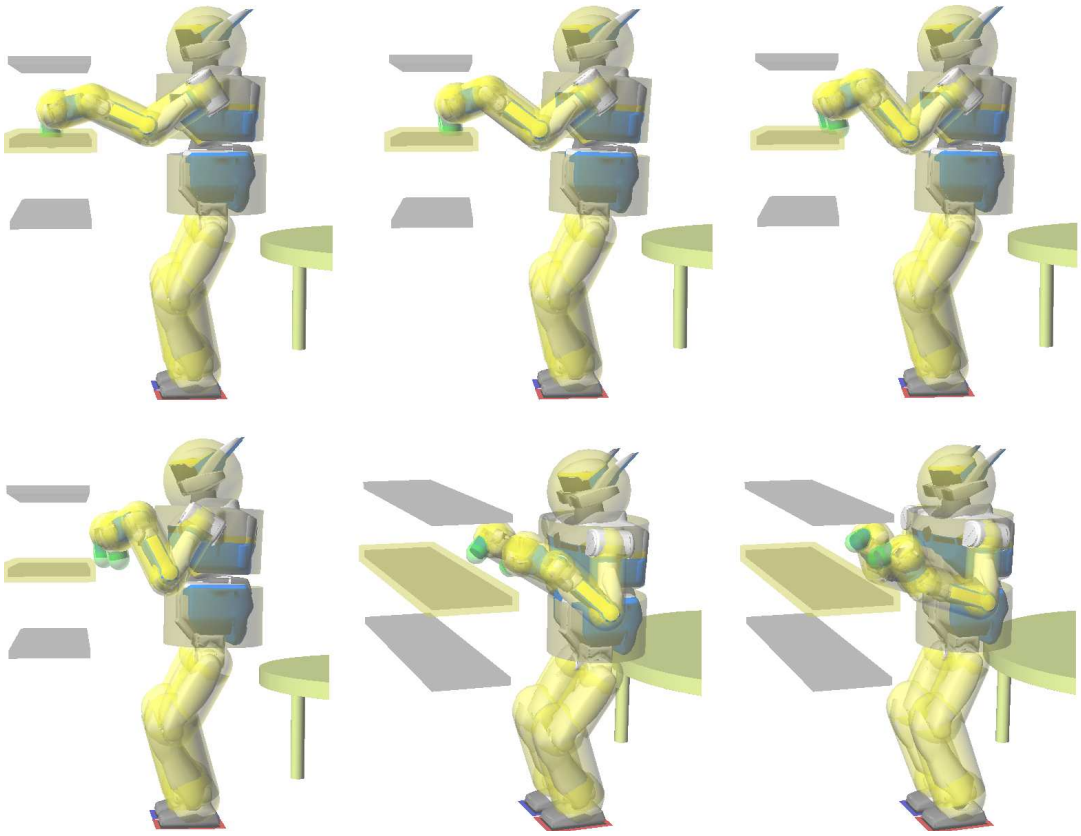


Figure 7.7: Phase 4: drawing back the hands. To avoid dragging the cups on the shelf, they are wrapped in a simplified geometry that purposefully intersects the shelf. A collision avoidance task (not constraint) is set in first priority such that, in absence of any other task, the cup is slightly pushed above the shelf. Position tasks (which can be replaced by plane inequality tasks to define target regions) are placed in second priority to draw the cups out.

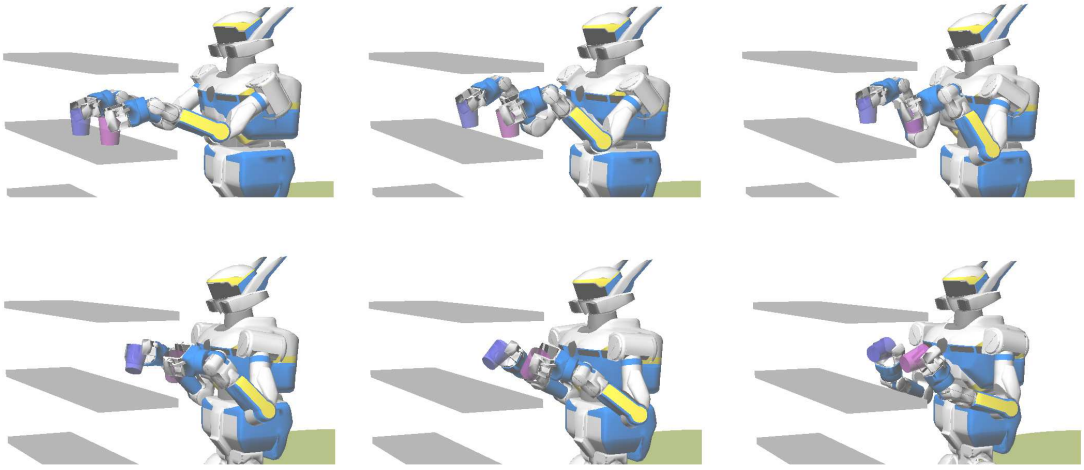


Figure 7.8: The cups end in an inclined orientation due to a configuration task always placed at lowest priority to regain a rest posture (as in figure 7.2) as much as possible.

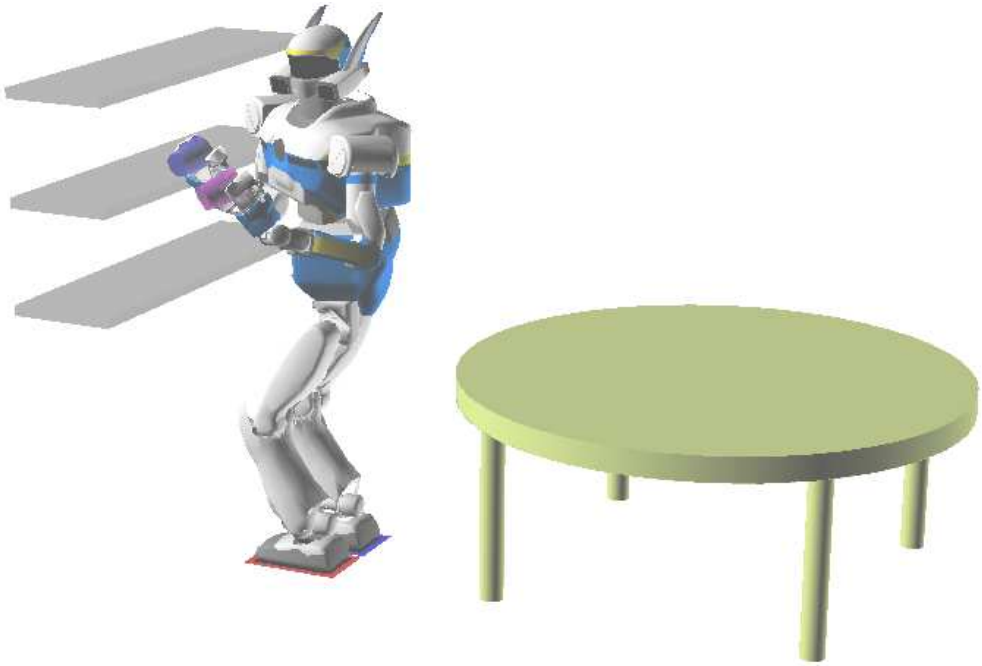


Figure 7.9: Phase 5: the robot is trapped in a local minimum with respect to its final task. The algorithm could not plan footsteps that bring the robot directly from the stance of figure 7.8 to one allowing to put the cups on the table or at least look at it. The space between the table and the shelves is too narrow.

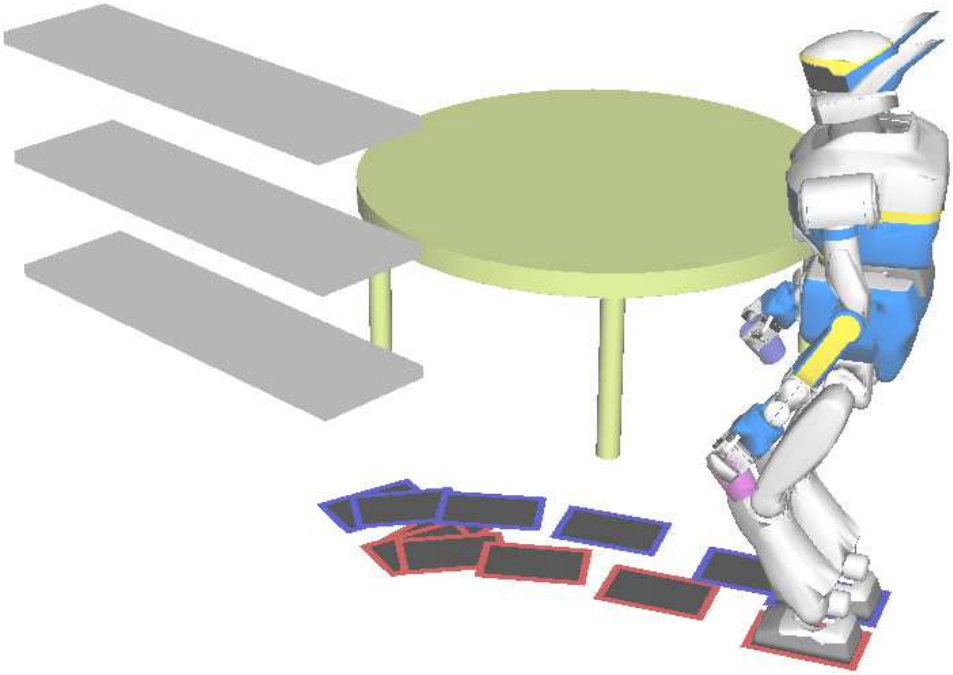


Figure 7.10: Phase 6: a higher layer of planning was required to escape eternal shelf contemplation. A random planner can give the robot an intermediary target before resuming usage of lower level algorithms.

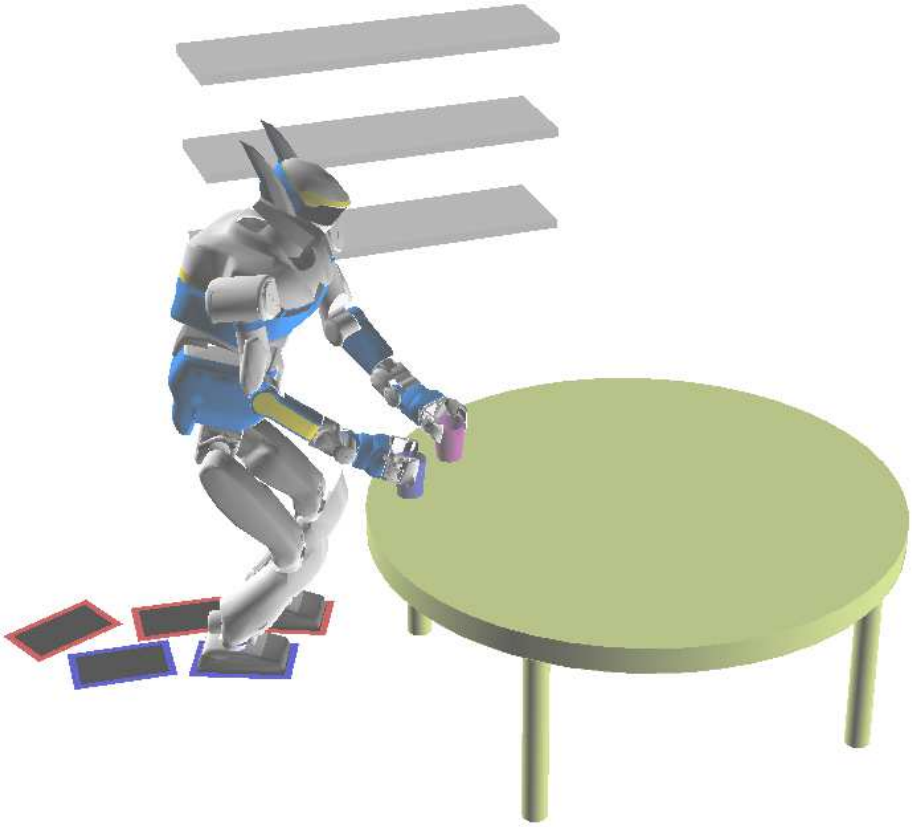


Figure 7.11: Phase 7: from the intermediary position the objective could be achieved.

8 Conclusion

Outcomes

We believe we have advanced towards the goals we set for the planning component of our humanoid robot. We designed a generic algorithm that relates a class of static tasks, functions of the configuration of the robot, to a path of footsteps. We rid ourselves from heuristical stepping strategies, which had to be customized to each task and each humanoid robot.

We looked into numerical frameworks for whole-body motion control and generalized the notion of static tasks to inequality functions of the robot configuration. We proposed to extend the formulation of strictly prioritized task-oriented kinematic control to static inequality tasks. Unlike the methods based on potential fields[Khatib 1986] where a hard inequality constraint is enforced by means of an equality constraint based on a measure of distance before constraint violation, we strived to characterize the set of controls which are rigorously defined by an inequality constraint. This lead us to characterize the sets of controls when both equality and inequality constraints are taken into account and consider the problem when prioritization is involved.

We were successful in developing an optimization algorithm that computes an optimal solution for a hierarchy of linear equality and/or inequality constraints while enforcing a strict priority order. We emphasize again the generality of this solver, presented in chapter 3 independently of the inverse kinematic setting.

We have investigated a dynamic method to generate dynamic walk motion for humanoid robots which we integrated within inverse kinematics frameworks, allowing simultaneous dynamic locomotion and

whole-body motion. Assembling all elements together, we obtained a powerful local motion controller that takes advantage of the rich mechanical system of the humanoid robot and its bipedal locomotion capabilities. While we relied on a kinematic control framework to show the principle of the approach, it can be applied in a dynamic control framework such as the one developed by [Khatib et al. 2004].

Limitations, extensions

The frameworks for local control present the advantage of low computation cost and easy goal description in work space, where the entire body's motion is driven by the desired motion and/or force of an end effector. Local controller are, however, prone to local minima.

In complement to the local motion control techniques there is a body of motion planning algorithms based on probabilistic search. The principle of these algorithms is to explore the configuration space to find a feasible, collision-free path of configurations linking an initial configuration of the robot to the configuration that makes its achieve a given task. One technique builds a *probabilistic roadmap* (PRM) [Kavraki et al. 1996] that is in fact a set of randomly sampled configurations, checked against collision, connected by paths of valid configurations which are calculated using a *local motion planning* algorithm. The more powerful the local motion planning algorithm, the less the configuration space is sampled. Our local footsteps planner may be used in this prospect. Instead of random sampling, one may use a diffusion process like a Rapidly-exploring Random Tree [Kuffner and LaValle 2000; Hsu et al. 1999] to sweep the configuration space from initial to desired configuration. Again, the connection between new diffusion nodes and previous ones is made by a local motion planning algorithm whose performance influences greatly the progression rate of the diffusion process. These search methods rely on a perfectly known environment, which may be subject to sudden changes that invalidate the current PRM/RRT from which a motion is being executed. Some works [Khatib et al. 2004; Brock and Khatib 2004; Khatib et al. 1998] have addressed this problem and highlighted the efficiency of local motion planning techniques for online trajectory revision while the global strategy is updated more slowly by the search algorithms.

In chapter 7, we illustrated a case where narrow space around the robot made the local footsteps planner fail to calculate a solution for the tasks at hand. Little bricks placed on the ground around the robot would have also caused the planner to fail since its feet are constrained to "slide" during the planning phase. [Kuffner et al. 2001] have addressed the problem of planning footsteps for a goal position in presence of obstacles that can be stepped over. Thee search from among a discrete set of plausible statically-stable single-step motions built as a decision tree from the start to the desired position for the robot. A general study of the coupling of search-based techniques with local minima was proposed by [Barraquand and Latombe 1991; Barraquand et al. 1992]. It is in a similar fashion that we view the integration of our footsteps local planner with probabilistic search algorithms in future works.

Thoughts on future work

The constraints and tasks shown in 4 were used throughout this work to design motion for the humanoid robot HRP-2, they can be viewed as a vocabulary for motion generation. If we construct a function that recognizes the motion of humans and translates it into a vocabulary of motion we will have made a significant step in the design of autonomous anthropomorphic systems. The recognition subfunction has been the focus of extensive research [Campbell and Bobick 1995; Aggarwal and Cai 1999; Wang et al. 2008; Horaud et al. 2009]. The adaptation to robots has gained attention lately as testify the works by [Suleiman et al. 2008] which proposed a method to transform Motion Capture data into dynamically stable motion for humanoid robots, or the works by [Billard et al. 2008; Pastor et al. 2009] on learning by demonstration and the work of [Kulic et al. 2008; Lee and Nakamura 2009] on recognition and mimesis of whole-body motion. These emerging motion acquisition methods are essential to the autonomy of humanoid robots as they would allow it to acquire previously non-encountered solutions to known problems and identify them on a basis of motion generation vocabulary. Combining this function with the inference of motion task semantics from observation should allow the robot to come with solutions for unknown problems involving motion and manipulation of objects. This is a vast and open research field to be applied to explored.

References

AGGARWAL, J. AND CAI, Q. 1999. Human motion analysis: A review. *Computer vision and image understanding 73,* 3, 428–440. 93

BAERLOCHER, P. 2001. Inverse kinematics techniques for the interactive posture control of articulated figures. Ph.D. thesis, Ecole Polytechnique Federale de Lausanne. 17

BAERLOCHER, P. AND BOULIC, R. 1998. Task-priority formulations for the kinematic control of highlyredundant articulated structures. In *IEEE-RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 323–329. 3

BARRAQUAND, J., LANGLOIS, B., AND LATOMBE, J.-C. 1992. Numerical Potential Field Techniques for Robot Path Planning. *22,* 2, 224–241. 92

BARRAQUAND, J. AND LATOMBE, J.-C. 1991. Robot Motion Planning: A Distributed Representation Approach. *10,* 6, 628–649. 92

BERTHOZ, A. 2002. *The brain's sense of movement*. Harvard University Press. 61

BILLARD, A., CALINON, S., DILLMANN, R., AND SCHAAL, S. 2008. Robot programming by demonstration. *Handbook of robotics*, 1371–1394. 93

BROCK, O. AND KHATIB, O. 2004. Elastic Strips: A Framework for Motion Generation in Human Environments. *21,* 12, 1031–1052. 92

CAMPBELL, L. AND BOBICK, A. 1995. Recognition of human body motion using phase space constraints. In *International Conference on Computer Vision*. Vol. 3. 93

CHAUMETTE, F. AND MARCHAND, E. 2001. A redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing. *IEEE Transactions on Robotics 17,* 5, 719–730. 3

CHESTNUTT, J. AND KUFFNER, J. 2004. A Tiered Planning Strategy for Biped Navigation. In *IEEE-RAS International Conference on Humanoid Robots*. Vol. 1. 422–436. 76

COLLETTE, C., MICAELLI, A., ANDRIOT, C., AND LEMERLE, P. 2008. Robust balance optimization control of humanoid robots with multiple non coplanar grasps and frictional contacts. In *IEEE International Conference on Robotics and Automation*. 3187–3193. 51

DECRÉ, W., SMITS, R., BRUYNINCKX, H., AND DE SCHUTTER, J. 2009. Extending iTaSC to support inequality constraints and non-instantaneous task specification. In *IEEE International Conference on Robotics and Automation*. 3

DIANKOV, R., RATLIFF, N., FERGUSON, D., SRINIVASA, S., AND KUFFNER, J. 2008. Bispace planning: Concurrent multi-space exploration. 4

ESCANDE, A., KHEDDAR, A., MIOSSEC, S., AND GARSAULT, S. 2009. Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2. *Experimental Robotics*, 293–302. 3

FAVERJON, B. AND TOURNASSOUD, P. 1987. A local based approach for path planning of manipulators with a high number of degrees of freedom. In *IEEE International Conference on Robotics and Automation*. Vol. 4. 1152–1159. 17, 39

FIACCO, A. AND MCCORMICK, G. 1987. *Nonlinear programming: sequential unconstrained minimization techniques*. Classics in Applied Mathematics. Society for Industrial Mathematics. 27

HOFMANN, A., POPOVIC, M., AND HERR, H. 2007. Exploiting angular momentum to enhance bipedal center-of-mass control. In *IEEE International Conference on Robotics and Automation*. 3

HORAUD, R., NISKANEN, M., DEWAELE, G., AND BOYER, E. 2009. Human Motion Tracking by Registering an Articulated Surface to 3D Points and Normals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 158–164. 93

HSU, D., LATOMBE, J., AND MOTWANI, R. 1999. Path planning in expansive configuration spaces. *9,* 4-5, 495–512. 92

KAJITA, S., KANEHIRO, F., KANEKO, K., FUJIWARA, K., HARADA, K., YOKOI, K., AND HIRUKAWA, H. 2003. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation*. Vol. 2. 1620–1626. 49, 51, 52, 72

KANEHIRO, F., LAMIRAUX, F., KANOUN, O., YOSHIDA, E., AND LAUMOND, J. 2008. A local collision avoidance method for non-strictly convex polyhedra. In *Robotics: Science and Systems Conference*. 39, 40

KANEHIRO, F., SULEIMAN, W., LAMIRAUX, F., YOSHIDA, E., AND LAUMOND, J. 2008. Integrating Dynamics into Motion Planning for Humanoid Robots. In *IEEE-RSJ International Conference on Intelligent Robots and Systems*. 660–667. 51

KANEKO, K., KANEHIRO, F., KAJITA, S., HIRUKAWA, H., KAWASAKI, T., HIRATA, M., AKACHI, K., AND ISOZUMI, T. 2004. Humanoid robot HRP-2. In *IEEE International Conference on Robotics and Automation*. Vol. 2. 1083–1090. 43

KAVRAKI, L., SVESTKA, P., LATOMBE, J., AND OVERMARS, M. 1996. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation 12,* 4, 566–580. 92

KHATIB, M., JAOUNI, H., CHATILA, R., AND LAUMOND, J.-P. 1998. *How to implement dynamic paths*. Lecture Notes in Control and Information Sciences, vol. 232. Springer. 92

KHATIB, O. 1986. The potential field approach and operational space formulation in robot control. *Adaptive and Learning Systems: Theory and Applications*, 367–377. 3, 91

KHATIB, O. 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation 3,* 1, 43–53. 51

KHATIB, O., BROCK, O., CHANG, K.-S., RUSPINI, D., SENTIS, L., AND VIJI, S. 2004. Human-Centered Robotics and Interactive Haptic Simulation. *23,* 2, 167–178. 92

KHATIB, O., SENTIS, L., PARK, J., AND WARREN, J. 2004. Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics 1,* 1, 29–43. 3, 92

KUFFNER, J., KAGAMI, S., NISHIWAKI, K., INABA, M., AND INOUE, H. 2002. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots 12,* 1, 105–118. 3

KUFFNER, J. AND LAVALLE, S. 2000. RRT-connect: An efficient approach to single-query path planning. In *IEEE Internatinal Conference on Robotics and Automation*. Vol. 2. 995–1001. 92

KUFFNER, J., NISHIWAKI, K., KAGAMI, S., INABA, M., AND INOUE, H. 2001. Footstep planning among obstacles for biped robots. In *IEEE-RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 500–505. 92

KULIC, D., TAKANO, W., AND NAKAMURA, Y. 2008. Incremental Learning, Clustering and Hierarchy Formation of Whole Body Motion Patterns using Adaptive Hidden Markov Chains. *International Journal of Robotics Research 27,* 7, 761–784. 93

LAMIRAUX, F. AND LAUMOND, J. 2000. Flatness and small-time controllability of multibody mobile robots: Application to motion planning. *IEEE Transactions on Automatic Control 45,* 10, 1878–1881. 60

LAUMOND, J. 2006. Kineo CAM: a success story of motion planning algorithms. *IEEE Robotics & Automation Magazine 13,* 2, 90–93. 44

LAWRENCE, C., ZHOU, J., AND TITS, A. 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. *Institute for Systems Research University of Maryland, College Park, MD 20742*. 6

LEE, D. AND NAKAMURA, Y. 2009. Mimesis Model from Partial Observations for a Humanoid Robot. *The International Journal of Robotics Research*. 93

LIÉGEOIS, A. 1977. Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms, IEEE trans. *IEEE Transactions on Systems, Man, and Cybernetics 7,* 12, 868–871. 3, 8

MANSARD, N. AND CHAUMETTE, F. 2007. Task Sequencing for High-Level Sensor-Based Control. *IEEE Transactions on Robotics 23,* 1, 60–72. 3

MANSARD, N., KHATIB, O., AND KHEDDAR, A. 2008. Integrating unilateral constraints inside the Stack Of Tasks. *IEEE Transactions on Robotics*. 3

MARCHAND, E. AND HAGER, G. 1998. Dynamic Sensor Planning in Visual Servoing. In *IEEE International Conference on Intelligent Robots and Systems*. 1988–1993. 3

NAKAMURA, Y. 1990. *Advanced robotics: redundancy and optimization*. Addison-Wesley Longman Publishing. 3, 8, 11, 13, 21

NAKAMURA, Y. AND HANAFUSA, H. 1986. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement, and control 108,* 3, 163–171. 11

NEO, E., YOKOI, K., KAJITA, S., AND TANIE, K. 2007. Whole-body motion generation integrating operator's intention and robot's autonomy in controlling humanoid robots. *IEEE Transactions on Robotics 23,* 4, 763–775. 3

PARK, K., CHANG, P., AND SALISBURY, J. 1997. A unified approach for local resolution of kinematic redundancy with inequality constraints and its application to nuclear power plant. In *IEEE International Conference on Robotics and Automation*. Vol. 1. 766–773. 44

PASTOR, P., HOFFMANN, H., ASFOUR, T., AND SCHAAL, S. 2009. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation*. 93

PEINADO, M., BOULIC, R., LE CALLENNEC, B., AND MEZIAT, D. 2005. Progressive cartesian inequality constraints for the inverse kinematic control of articulated chains. *Eurographics*, 93–96. 17

SALINI, J., BARTHÉLEMY, S., AND BIDAUD, P. 2009. LQP controller design for generic whole body motion. In *Climbing and Walking Robots*. 3

SHADMEHR, R. AND WISE, S. 2005. *The computational neurobiology of reaching and pointing: a foundation for motor learning*. Bradford Books, Computational Neuroscience. MIT Press. 56

SICILIANO, B. AND SLOTINE, J. 1991. A general framework for managing multiple tasks in highly redundantrobotic systems. In *International Conference on Advanced Robotics*. 1211–1216. 3, 21

SREENIVASA, M., SOUERES, P., LAUMOND, J.-P., AND BERTHOZ, A. 2009. Steering a humanoid robot by its head. In *IEEE International Conference on Intelligent Robots and Systems*. 4451–4456. 61, 72

SUGIHARA, T., NAKAMURA, Y., AND INOUE, H. 2002. Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control. In *IEEE International Conference on Robotics and Automation*. Vol. 2. 1404– 1409. 56

SULEIMAN, W., YOSHIDA, E., KANEHIRO, F., LAUMOND, J., AND MONIN, A. 2008. On human motion imitation by humanoid robot. In *IEEE International Conference on Robotics and Automation*. 2697–2704. 93

VUKOBRATOVIĆ, M., BOROVAC, B., SURLA, D., AND STOKIC, D. 1990. *Biped locomotion: dynamics, stability, control, and application*. Scientific Fundamentals of Robotics. Springer. 51

WANG, J., FLEET, D., AND HERTZMANN, A. 2008. Gaussian process dynamical models for human motion. *IEEE Transaction on Pattern Analysis and Machine Intelligence 30,* 2, 283–298. 93

WIEBER, P. 2002. On the stability of walking systems. In *International Workshop Humanoid Human Friendly Robotics*. 53–59. 50

WOLF, A. 1996. Transforming ordered constraint hierarchies into ordinary constraint systems. *Lecture Notes in Computer Science*, 171–188. 27

YOSHIDA, E., KANOUN, O., ESTEVES, C., AND LAUMOND, J. 2006. Task-driven support polygon reshaping for humanoids. In *IEEE-RAS International Conference on Humanoid Robots*. 208–213. 3, 55, 57, 58

YOSHIDA, E., MALLET, A., LAMIRAUX, F., KANOUN, O., STASSE, O., POIRIER, M., DOMINEY, P., LAUMOND, J., AND YOKOI, K. 2007. "Give me the purple ball" he said to hrp-2 n. 14. In *IEEE-RAS International Conference on Humanoid Robots*. 89–95. 4, 58

YOSHIDA, E., POIRIER, M., LAUMOND, J., ALAMI, R., AND YOKOI, K. 2007. Pivoting based manipulation by humanoids: a controllability analysis. In *IEEE-RSJ International Conference on Intelligent Robots and Systems*. 1130–1135. 4, 60

YOSHIDA, E., POIRIER, M., LAUMOND, J., KANOUN, O., LAMIRAUX, F., ALAMI, R., AND YOKOI, K. 2008. Whole-body motion planning for pivoting based manipulation by humanoids. In *IEEE International Conference on Robotics and Automation*. 3181–3186. 60, 76

ZHAO, J. AND BADLER, N. 1994. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics 13,* 4, 336. 3, 17

Auteur: Oussama KANOUN

Titre: Contribution á la planification de mouvement pour robots humanoïdes

Résumé de la thèse: cette thèse porte sur des algorithmes de contrôle et de planification de mouvements pour les robots humanoïdes. Le grand nombre de paramètres caractérisant ces systèmes a conduit au développement de méthodes numériques, d'abord appliquées aux bras manipulateurs et récemment adaptées pour les structures plus complexes. On relève particulièrement les formalismes de commande cinématique et dynamique par priorité qui permettent de produire un mouvement selon une hiérarchie pré-établie des tâches. Au cours de ce travail, nous avons identifié le besoin d'étendre ce formalisme afin de tenir compte de contraintes unilatérales. Nous nous sommes par ailleurs intéressés à la planification de la locomotion en fonction des tâches. Nous proposons une modélisation jointe du robot et de sa trajectoire de marche comme une structure articulée unique saisissant à la fois les degrés de liberté actionnés (articulations motorisées du robot) et non-actionnés (positionnement absolu dans l'espace). L'ensemble de ces algorithmes, qui seront longuement illustrés, ont été implémentés au sein du projet HPP (Humanoid Path Planner) et validés sur le robot humanoïde HRP-2.

Mots clés: cinématique inverse, contrôle optimal, optimisation sous contraintes, priorité, locomotion bipède, robot humanoïde, planification de mouvement

Title: Task-driven motion control for humanoid robots

Abstract of the thesis: this thesis is related to motion control and planning algorithms for humanoid robots. For such highly-parameterized systems, numerical methods are well adapted and have thus been the center of increasing attention in the recent years. Among the prominent numerical schemes, we recognized the prioritized inverse kinematics and dynamics frameworks to hold key features to plan motion for humanoid robots, such as the possibility to control the motion while enforcing a strict priority order among tasks. We have, however, identified a lack of support of strict priority enforcement when inequality constraints are to be accounted for in the numerical schemes and we were successful in proposing a solution to this shortcoming. We also considered the problem of planning bipedal locomotion according to any given tasks. We proposed to model this problem as an inverse kinematics problem, by considering the kinematic structure of the robot and its walk path as a single unified structure that captures both the degrees of freedom of the robot which are actuated (motorized joints) and those which are not (position and orientation in space). The presented algorithms, which will be abundantly illustrated, have been implemented within the HPP (Humanoid Path Planner) project and validated on the humanoid robot HRP-2.

Keywords: inverse kinematics, optimal control, constrained optimization, priority, bipedal locomotion, humanoid robot, motion planning