Synthesis of Complex Humanoid Whole-Body Behavior: a Focus on Sequencing and Tasks Transitions

Joseph Salini, Vincent Padois, Philippe Bidaud

Synthesis of Complex Humanoid Whole-Body Behavior: a Focus on Sequencing and Tasks Transitions

Joseph Salini, Vincent Padois and Philippe Bidaud

Abstract— **We present a novel approach to deal with transitions while performing a sequence of dynamic tasks with a humanoid robot. The simultaneous achievement of several tasks cannot be ensured, so we use a strategy based on weights to represent their relative importance. The robot interacts with a changing environment, and the input torques are different depending on whether the robot performs tasks in a constrained state (e.g. in contact) or not. We develop a solution with smooth weights variations and transitional tasks which avoids sharp torque evolutions. In order to validate this approach, simulations are carried out on a virtual iCub robot which is assigned the realization of a complex mission involving various changing tasks.**

I. Introduction

Challenges in the control of complex robotic systems are evolving from the derivation of efficient controllers dedicated to elementary tasks such as walking ([1],[2]) or multiple contact control ([3]) to the blending of such controllers within higher level control architectures for dealing with complex activities.

These new control architectures are expected to provide complex robotic systems with the ability to reactively perform sequences of complex tasks in unstructured environments and in dynamically changing contexts where unexpected physical interactions may occur between the robot and its environment. Besides, achieving complex behaviors may also lead to perform several, potentially antagonistic tasks at the same time, while having critical importance.

Humanoid robots (physical ones or avatars) are good examples of such complex systems. As a matter of fact the scientific literature in this domain recently proposed control architectures to tackle these new challenges.

Among this literature, the work of Sentis *et al.* ([4], [5]) provides analytical methods to solve the humanoid control problem in cases where walking and compliant reaching tasks are involved while constraints such as equilibrium, collision and joint limits avoidance have to be satisfied. Tasks are hierarchically achieved and the proposed framework fully takes into account dynamic couplings. While elegant and physically meaningful, this type of method leads to a rather complex problem formulation which induces an extensive use of computationally expensive inversion techniques. Also this approach supposes the definition of a tasks hierarchy and, most importantly, it does not allow to account for unilateral constraints (contacts for example) in a straightforward way.

In order to obtain more generic control problem formulations as well as to take unilateral constraints into account simply, optimization techniques have recently been used extensively in the robotic context. More specifically, Linear Quadratic Programming (LQP) has been used in different contexts. For example, Abe *et al.* ([6]) make use of LQP in the framework of animation of humanoid avatars achieving various tasks. Colette *et al.* ([7]) pursue the same goal, using similar tools with a focus on robust balancing. The same kind of approach is being used in the work of Escande *et al.* ([8]) at the inverse velocity kinematics level with efficient implementation on a humanoid robot in mind and allowing to enforce priorities both at the tasks and constraints levels. Finally, even though not applied to the humanoid case directly, the work of Decré *et al* ([9]) makes use of cascading LQPs to formulate complex control problems with priorities between tasks as well as unilateral constraints.

Even though very promising, the optimization approach to control as it is implemented in these recent contributions still lacks two interesting features. The first one is the possibility to resort to soft hierarchies of tasks. Indeed, tasks hierarchy is of high interest but it is not always simple to define strict priorities between tasks especially in highly varying contexts where there is no guarantee that all tasks can be achieved simultaneously. Secondly, keeping the implementation of these controllers on real humanoid robots in mind, control input smoothing and saturation is very important and sequences of tasks of different nature (*contact* $\rightarrow$ *free-space* is one example) should ensure soft transitions, being them planned or reactively activated.

As a tentative to propose a control framework addressing these two challenges, the work presented in this paper is an extension of our previous work ([10]). We propose the derivation of a humanoid robot controller based on LQP that allows the description of pre-planned tasks sequences including transition conditions and authorizing the reactive insertion of new tasks to ensure a proper behavior of the system in dynamically changing environments. Several tasks can be specified simultaneously and achieved based on a soft hierarchy implemented through a generalized task weighting strategy. This weighting strategy is also used when a transition occurs in order to ensure smoothed, realistic control inputs.

The paper is organized as follows. In the first section, the formulation of control objectives in terms of operational

All authors are with:
Université Pierre et Marie Curie
Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222
Pyramide Tour 55, Boite Courrier 173
4 Place Jussieu
F-75005 Paris
email: {salini,padois,bidaud}@isir.upmc.fr

accelerations and forces is described and task servoing is discussed. Higher level tasks are are also introduced among which impedance and predictive ZMP control. Then, in the second section, the retained constraints are presented and an emphasis is put on unilateral contact constraints. The third section synthesizes the LQP controller based on the chosen tasks and constraints formulation. Results on a simulated iCub robot are then proposed in the fourth section through the presentation of different scenarii. These results are discussed in conclusion where some insights on possible future work are also given.

II. Control objectives

A humanoid robot can be modeled as an under-actuated tree-structure composed of rigid bodies linked together with revolute joints. The equation of motion of such a system can be derived from the Euler-Lagrange formalism

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + N(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{g}(\boldsymbol{q}) + S\boldsymbol{\tau} + J_c(\boldsymbol{q})^t \boldsymbol{w}_c \tag{1}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are respectively the generalized coordinates, velocity and acceleration vectors, $M, N, \boldsymbol{g}$ the generalized inertia matrix, the Coriolis and non-linear effects matrix and the gravity vector, $S, \boldsymbol{\tau}$ the actuation matrix and the input torque vector, and finally $J_c, \boldsymbol{w}_c$ the contact points Jacobian and the contact wrench. S allows to account for the fact that some degrees of freedom are not actuated (typically the free-floating base). This equation is nonlinear and generally linearized around the state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $M, N, \boldsymbol{g}, S, J_c$ are thus supposed to be known. As a matter of fact, a task can either be described at the acceleration level $(\ddot{\boldsymbol{q}})$ or at the wrench level $(\boldsymbol{w}_c)$, $\boldsymbol{\tau}$ being the control variable of the system. In the remaining of this paper, $\boldsymbol{\chi} = [\ddot{\boldsymbol{q}}, \boldsymbol{w}_c, \boldsymbol{\tau}]^T$ is called the *action variable* of the system.

A. Generic Task Definition

A task can be defined as the servoing of a specific 3D frame attached to the robot body to a desired goal value. This control objective can either be described in terms of twist (denoted $\boldsymbol{t}$) and/or wrench ([11]). In the case of a twist (velocity) task, one needs to relate $\boldsymbol{t}$ to $\ddot{\boldsymbol{q}}$ which can be done using the task jacobian J

$$\dot{\boldsymbol{t}} = J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}. \tag{2}$$

It can also be described in term of a wrench and one needs to relate $\boldsymbol{w}$ to $\boldsymbol{\tau}$ which can similarly be done using the dual relation

$$\boldsymbol{\tau} = J(\boldsymbol{q})^t \boldsymbol{w}. \tag{3}$$

Generally, the achievement of the task cannot be guaranteed because of the constraints acting on the system. This leads to define a task as an error to minimize with respect to a desired goal. Thus, a task can be written as

$$\min \|\boldsymbol{x}^{des} - \boldsymbol{x}\| \tag{4}$$

where $\boldsymbol{x}$ can represent a set of accelerations $\dot{\boldsymbol{t}}$ and/or wrenches $\boldsymbol{w}$ and the superscript ^{des} stands for *desired*.

This formulation allows to describe cartesian space tasks: reaching, obstacle avoidance, Center of Mass (CoM) control, contact force control, etc. Joint space tasks (posture control, joint limit avoidance, torque minimization, etc.) can easily be described using the same formalism, the corresponding jacobian matrices becoming trivial. This multiplicity of tasks is illustrated on Figure 1.

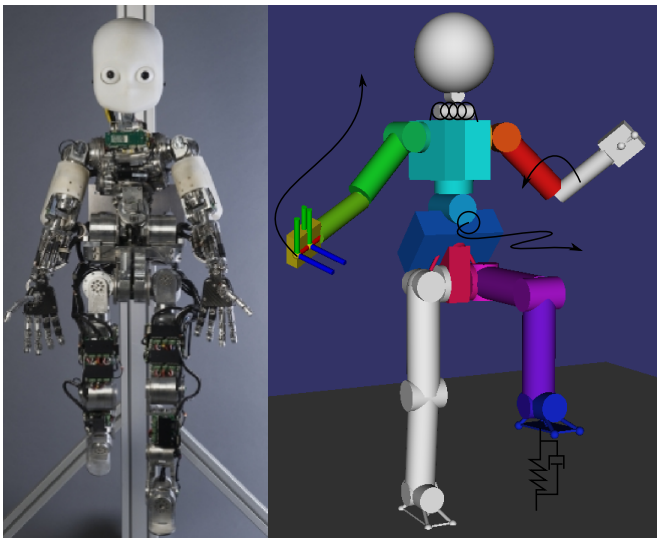

Fig. 1. Illustration of real and virtual iCub robots with prescribed tasks of different natures.

B. Task Servoing

In order to ensure some robustness with respect to the robot model, as well as to enforce a given task behavior (e.g. a virtual spring-damper system), a controller can be derived at the task level. When dealing with twists during a tracking task, this type of controller is often written

$$\dot{\boldsymbol{t}}^{des} = \dot{\boldsymbol{t}}^{goal} + K_p \boldsymbol{\epsilon}_p + K_d \dot{\boldsymbol{\epsilon}}_p \tag{5}$$

where $\boldsymbol{\epsilon}_p$ is the pose error which can be trivially computed when dealing with position and requires to resort to a non-minimal representation of the orientation such as quaternions ([12]). Reaching is a particular case of the tracking task where the goal trajectory is reduced to one point and $\dot{\boldsymbol{t}}^{goal} = \boldsymbol{t}^{goal} = \boldsymbol{0}$.

An equivalent wrench formulation leads to

$$\boldsymbol{w}^{des} = \boldsymbol{w}^{goal} + K_{fp}\boldsymbol{\epsilon}_w + K_{fi}\int \boldsymbol{\epsilon}_w dt \tag{6}$$

where $\boldsymbol{\epsilon}_w$ is the wrench error.

C. Higher level tasks

1) Impedance Controller: Impedance control is a way to unify twist and wrench control ([13]). Instead of directly controlling forces or accelerations, impedance control rather aims at inducing a proper behavior of the system with respect to contacts with its environment. A typical approach consists in setting a reference position to a frame attached to the robot and to control the reaction wrench induced by a disturbance.

It allows the use of a unified controller for non-constrained and constrained motions, e.g. when in contact.

A simple impedance controller can be modeled using a spring-damper system linking the controlled part of the system to a virtual reference. Hence, the choice of the reference position, the stiffness and damping coefficients determines the behavior of the reaction which may change, for instance to speed up the completion of the task, or to increase the contact wrench of the end-effector at contact point.

2) Zero Moment Point Controller: An approximation of the Zero Moment Point (ZMP) ([1]) can be controlled in order to perform a walking or a balancing task, through the control of the Center of Mass (CoM). It is computed as follow

$$\begin{aligned} \ddot{\boldsymbol{c}} &= J_{com}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}_{com}(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \boldsymbol{z} &= \boldsymbol{c} - \frac{h}{g}\ddot{\boldsymbol{c}} \end{aligned} \tag{7}$$

where J_{com} is the jacobian of the CoM, $\boldsymbol{z}, \boldsymbol{c}$ are respectively the horizontal position of the ZMP and of the CoM, h is the height of the CoM and g is the gravity value. The direct control of $\boldsymbol{z}$ may be unstable, so an approach which takes into account the prediction of the future position must be implemented. As described in [14], the initial configuration of the CoM is denoted $C_0 = [\boldsymbol{c}, \dot{\boldsymbol{c}}, \ddot{\boldsymbol{c}}]^T$ and the prediction of the ZMP along a time horizon H is denoted $Z_H = [\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_H]^T$. Hence, the input matrix is composed of the future CoM jerks $\dddot{C}_H = [\dddot{\boldsymbol{c}}_0, \dddot{\boldsymbol{c}}_1, ..., \dddot{\boldsymbol{c}}_{H-1}]^T$ and one can compute the state and input matrices P_x, P_u to obtain the relation

$$Z_H = P_x C_0 + P_u \dddot{C}_H$$

The goal is to minimize the norm of the difference between a ZMP reference (Z_{ref} computed a priori) and the predicted trajectory. This minimization problem is unconstrained, so it can be solved directly, which gives

$$\dddot{c}_H = -(P_u^T.P_u + R.I_N)^{-1}.P_u^T.(P_x.X_{com} - Z_{ref})$$

where R is a ratio between the state and the command error, and I_N is the identity matrix. The first row of this input matrix is used to control the CoM and ZMP trajectories.

III. Constraints

The tasks completion is bounded by the environment of the robot and by its own technological limits. This section presents the major constraints faced by the system, but this list is not exhaustive and can be extended with custom constraints.

A. Equation of motion

The first constraint is the one due to the law of physics which imposes the evolution of $\boldsymbol{\chi}$ according to equation (1).

B. Technological limits

The characteristics of the actuators bound their range of action. Generally, the range of motion, velocity, acceleration and torque are bounded

$$\boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} \tag{8}$$
$$\ddot{\boldsymbol{q}}_{min} \leq \ddot{\boldsymbol{q}} \leq \ddot{\boldsymbol{q}}_{max} \tag{9}$$
$$\dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_{max} \tag{10}$$
$$\boldsymbol{q}_{min} \leq \boldsymbol{q} \leq \boldsymbol{q}_{max} \tag{11}$$

In order to describe all constraints in terms of $\boldsymbol{\chi}$, the action variable of the system, constraints (10) and (11) are respectively replaced by

$$\dot{\boldsymbol{q}}_{min} \leq \dot{\boldsymbol{q}} + \ddot{\boldsymbol{q}} h_2 \leq \dot{\boldsymbol{q}}_{max} \tag{12}$$
$$\boldsymbol{q}_{min} \leq \boldsymbol{q} + \dot{\boldsymbol{q}} h_1 + \ddot{\boldsymbol{q}}(h_1)^2/2 \leq \boldsymbol{q}_{max} \tag{13}$$

where h_1 and h_2 are anticipation coefficients set to predict the future value of the state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ given the generalized acceleration $\ddot{\boldsymbol{q}}$ of the system.

C. Interaction

1) Frictional Contact: The robot interacts with its environment through a set of contact points linked to its bodies. Restricting, for simplicity, twists and wrenches to their linear velocities and forces components respectively, each contact point i can be described by its linear velocity $\boldsymbol{v}_{ci} = J_{ci}\dot{\boldsymbol{q}} \in \mathbb{R}^3$ where J_{ci} is the jacobian[1] of the contact point and by its force denoted $\boldsymbol{f}_{ci} \in \mathbb{R}^3$. Several types of interaction can be imagined such as frictional contact, ball and socket, sliding, etc. Focusing on frictional contacts, four cases have to be distinguished:

- c_1 the contact is persistent, $\boldsymbol{v}_{ci} = \boldsymbol{0}$ and $\boldsymbol{f}_{ci}$ lies inside the Coulomb cone;
- c_2 the contact is lifting, $\boldsymbol{v}_{ci}.\boldsymbol{n} > 0$ and $\boldsymbol{f}_{ci} = \boldsymbol{0}$;
- c_3 there is no contact, $\boldsymbol{v}_{ci} \in \mathbb{R}^3$ and $\boldsymbol{f}_{ci} = \boldsymbol{0}$;
- c_4 the contact is sliding $\boldsymbol{v}_{ci} \times \boldsymbol{n} \neq \boldsymbol{0}$ and $\boldsymbol{f}_{ci} \neq \boldsymbol{0}$;

where $\boldsymbol{n}$ is the normal vector of contact.

The contact force can be decomposed into two components, a normal force $\boldsymbol{fn}_{ci} = \boldsymbol{f}_{ci} \cdot \boldsymbol{n}$ and a tangential force $\boldsymbol{ft}_{ci} = \boldsymbol{f}_{ci} - (\boldsymbol{f}_{ci} \cdot \boldsymbol{n})\boldsymbol{n}$. Hence, the Coulomb cone constraint is written:

$$\|\boldsymbol{ft}_{ci}\| \leq \mu_i \|\boldsymbol{fn}_{ci}\| \tag{14}$$

where μ_i is the friction coefficient of contact i. A polygonal approximation of the cone, lying in the real Coulomb cone, may be used to linearize equation (14). Given C_{fi} the representation of this linearized cone, the inequality becomes

$$C_{fi}\boldsymbol{f}_{ci} \leq \boldsymbol{0} \quad \forall i. \tag{15}$$

Within the framework of this work, only the two first cases are treated. The third one is trivially taken into account without writing any specific constraint equation. The fourth one is out of scope for this work.

[1] dependences to $\boldsymbol{q}$ are dropped here for the sake of clarity

2) Kinematic-Loop: A kinematic loop links two points in the tree-structure of the system, which results in the following equality on their respective joint velocities. Thus, considering each kinematic loop j, we have

$$(J_{j1} - J_{j2})\dot{\boldsymbol{q}} = \boldsymbol{0} \quad \forall j. \tag{16}$$

Again, these equations do not directly depend on the action variable $\boldsymbol{\chi}$ described in section II. Rewriting these equations, we get the following interaction constraints equations

$$c_1 \quad \begin{cases} \boldsymbol{v}_{ci_p} = \boldsymbol{0} \ : J_{ci_p}\ddot{\boldsymbol{q}} + \dot{J}_{ci_p}\dot{\boldsymbol{q}} = \boldsymbol{0} \\ C_{fi_p}\boldsymbol{f}_{ci_p} \leq \boldsymbol{0} \end{cases} \quad \forall i_p \tag{17}$$

$$c_2 \quad \begin{cases} \boldsymbol{v}_{ci_l}.\boldsymbol{n} > \boldsymbol{0} \ : J_{ci_l}\ddot{\boldsymbol{q}} + \dot{J}_{ci_l}\dot{\boldsymbol{q}}.\boldsymbol{n} > \boldsymbol{0} \\ \boldsymbol{f}_{ci_l} = \boldsymbol{0} \end{cases} \quad \forall i_l \tag{18}$$

$$(J_{j1} - J_{j2})\ddot{\boldsymbol{q}} + (\dot{J}_{j1} - \dot{J}_{j2})\dot{\boldsymbol{q}} = \boldsymbol{0} \quad \forall j \tag{19}$$

where i_p and i_l respectively describe persistent and lifting contacts.

IV. Optimization problem formulation

Tasks and constraints being formalized as a set of functions to minimize and as a set of inequalities respectively, optimization techniques offer a strong and very general mathematical framework to solve the control problem. Given the quadratic form of the tasks and the linearity of the constraints described in sections II and III, Linear Quadratic Program (LQP) is a good candidate, especially because of its computational efficiency. While computationally more costly, Second Order Cone Program (SOCP) could also be used to handle constraints related to the friction cones without simplification. The implementation of LQP done in this work could easily be converted into a SOCP ([15]).

A. Generic LQP

A LQP solves the following problem

$$\begin{aligned} \min_{(\boldsymbol{\chi})} \quad & \tfrac{1}{2}\left(\boldsymbol{\chi}^T P \boldsymbol{\chi} + \boldsymbol{p}^T \boldsymbol{\chi}\right) \\ s.t.: \quad & G\boldsymbol{\chi} \preceq \boldsymbol{h} \\ & A\boldsymbol{\chi} = \boldsymbol{b} \end{aligned} \tag{20}$$

where $\boldsymbol{\chi}$ is the vector to optimize, $P, \boldsymbol{p}$ represent the quadratic cost function, $G, \boldsymbol{h}$ define the inequality constraints, and $A, \boldsymbol{b}$ define the equality constraints.

As mentioned in section II, tasks are described as a norm to minimize which is a function of $\ddot{\boldsymbol{q}}$, $\boldsymbol{w}_c$ or $\boldsymbol{\tau}$, depending on their nature. Having introduced $\boldsymbol{\chi}$ as the concatenation of these three variables, one can, without loss of generality formulate any task k in the following manner

$$T_k(\boldsymbol{\chi}) = \|E_k\boldsymbol{\chi} + \boldsymbol{f}_k\|^2 \tag{21}$$

where E_k and $\boldsymbol{f}_k$ are properly computed to represent the respective error to minimize, and the norm is squared to fit the LQP. For instance, considering the case of task inducing the control of the acceleration of one end-effector (EF) of the robot, we have $E = [J_{EF}\ 0\ 0]$ and $\boldsymbol{f}_{EF} = \dot{J}_{EF}\dot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}^{des}$.

As for the constraints, they are given by equations (1), (8), (9), (12), (13), (17), (18) and (19). It is important to notice that these constraints may not always be compatible inducing an empty set of solutions. This is not acceptable in the context of real time robot control. To solve this difficulty, a first strategy would require to resort to constraints relaxation. This does not really make sense given the type of problem solved. A second strategy consists in rewriting the constraints set in order to ensure compatibility and thus the existence of at least one solution. This second strategy was not implemented within the framework of this work but interested reader can refer to the recent work of Rubrecht *et al.* ([16]) who treat this problem at the velocity kinematics level.

B. Tasks priorities: Hierarchy vs Weighting

When two or more tasks have to be performed at the same time, conflicting situations may occur. Some tasks must have more importance to ensure their achievement at the expenses of other ones. Two strategies are possible which are either based on a strict hierarchy or relative weighting between tasks. Before giving more details about these two approaches, it is important to notice here that in both cases, tasks may not be compatible with the constraints or tasks may be incompatible one another, in which case they cannot be achieved perfectly.

1) Hierarchy: To enforce such priorities, the most common strategy is to resort to strict hierarchies of tasks ([17]). A strict hierarchy is an ordered list of tasks where the lower ones cannot interfere with the higher ones. This solution is relatively straightforward and just requires relative orders. From an algorithmic point of view, strict hierarchies require the projection of low priority tasks jacobians in the null-space of higher priority tasks jacobians.

In this case, the optimization problem is solved recursively. Given a ordered set of n tasks $T_i(\boldsymbol{\chi}) = \|E_i\boldsymbol{\chi} + \boldsymbol{f}_i\|^2$, the algorithm becomes

`for i=1..n:`

$$\begin{aligned} \min_{(\boldsymbol{\chi})} \quad & \frac{1}{2}\left((T_i(\boldsymbol{\chi}))^2 + (w_0 T_0)^2\right) \\ s.t.: \quad & G\boldsymbol{\chi} \preceq \boldsymbol{h} \\ & A_i\boldsymbol{\chi} = \boldsymbol{b}_i \end{aligned}$$

$$\begin{aligned} A_{i+1} &\leftarrow A_i \cup E_i \\ \boldsymbol{b}_{i+1} &\leftarrow \boldsymbol{b}_i \cup (E_i\boldsymbol{\chi}_i^*) \end{aligned}$$

where $\boldsymbol{\chi}_i^*$ is the solution of program i, $A_1 = A$, $\boldsymbol{b}_1 = \boldsymbol{b}$ and T_0 is a task dedicated to the minimization of the whole optimization variable. This is required when the control problem has many solutions in order to ensure uniqueness and more specifically to provide a solution that minimizes accelerations, contact forces and actuation torques. This task is given, using a weight w_0, a very small importance compared to others.

While providing convincing results, this priority strategy has several drawbacks. The first one lies in the fact that this algorithm solves as many LQP as the number of tasks per call which increases the overall computation time. Moreover

the extension of the constraints set at each step increases the risks of not finding any solution including the complete set of tasks. In other words, low priority tasks may not be achieved at all. The second disadvantage of this method is that it requires to define relative orders between tasks which is never obvious in most robotics contexts. Moreover, this ordered list is not always feasible (e.g. a loop in the relative orders), and a transition or a replacement in the stack of tasks may involve discontinuities in the control signal, which must be avoided on real systems.

2) Weighting: The approach retained in this paper relies on weights. This weighting strategy associates each task with a coefficient thats sets its importance with respect to others (a task with a very high relative weight gets the highest priority). As a consequence priorities are not strict and all tasks are achieved according to the trade-off defined by the weights.

Given a set of tasks with their relative weights (T_i, w_i) $i \in [1..n]$, the LQP becomes:

$$\begin{array}{ll} \min\limits_{(\boldsymbol{\chi})} & \frac{1}{2}\left(\sum_i((w_i.T_i(\boldsymbol{\chi}))^2) + (w_0 T_0)^2\right) \\ s.t.: & G\boldsymbol{\chi} \preceq \boldsymbol{h} \\ & A\boldsymbol{\chi} = \boldsymbol{b} \end{array}$$

where $w_0 \ll w_i \ \forall i$. This program is solved only one time per call and the time of computation depends mainly on the dimension of the set of constraints. Practically speaking, a task can be considered much more important if its weight is ten times higher than the weights of other tasks.

Even though defining the value of the weights for all tasks may not be a simple problem, it leaves more flexibility and lower priority tasks are never completely abandoned even in the worst cases. Also, similarly to the previous strategy, if an exact solution exists (i.e. a solution that minimizes exactly all tasks), it will be found, i.e. all tasks will be perfectly achieved. Finally, this method allows to change the weights continuously in order to generate smooth transitions. This is an important feature when facing the implementation on a real system, especially when dealing with complex robotic missions where tasks can be reactively activated based on the environment dynamics.

3) Transitions: The knowledge of the past solution is not given to the optimization program, so discontinuities may happen if two successive problems are not close enough. Sharp transitions in $\boldsymbol{\chi}$ are obviously not desirable at least from a practical point of view (e.g. a sharp transition in terms of contact wrench $\boldsymbol{w}_c$ would either mean a sudden lost of support or a strong impact).

These discontinuities are very likely to occur in the case where the relative importance of tasks is suddenly modified or when constraints appear or disappear. To deal with such discontinuities, one can

- add a constraint related to the continuity of the control signal (e.g. bound the values of the derivative of the action variable) – this possibility has not been implemented in this paper;
- ensure continuity using a properly weighted pseudo inverse jacobian as proposed in [18] – this is not possible when using optimization-based control approaches (e.g. LQP) since no inverse is computed explicitly;
- ensure continuity using a continuous variation of the weights associated to each task – this is the approach retained in this paper.

Indeed, continuous change of the tasks weights should imply smooth transition. Besides, a planned transition from a constraint state to another one (for instance the lost of support of a chair when standing) can be achieved with the use of a transitional task whose weight gains importance during a short period of time.

Of course, the choice of the tasks weights and their evolutions should be automated to generate an efficient generic whole-body motion controller. Here, the authors use a relatively sequential approach. Each task has triggers for its start and its end. A supervisor evaluates its completeness and modifies its weight according to the mission context and with respect to the criticality of the other tasks[2]. As a matter of fact, the tasks weights can be modified automatically without requiring any complex tuning

$$w_i = \begin{cases} w_{prev}, \ t < t_e \\ \exp\left(\frac{\log(w_{next}) - \log(w_{prev})}{t_s - t_e} t\right), \ t \in [t_e, t_s] \\ w_{next}, \ t > t_s \end{cases} \quad (22)$$

where w_{prev} and w_{next} are the weights value of task i respectively for the previous and the next mission contexts, t_e and t_s are the trigger time respectively for the end of the previous mission context and the start of the next one. The time between t_e and t_s is thus naturally defined as the transition time.

This should lead to a good overall behavior even if complex cases.

V. Experiments & Results

The experiments have been carried out using Arboris-Python ([19]), an open-source dynamic simulator developed at ISIR with the Python programming language. The implementation of LQP relies on CVXOPT/CVXMOD, two free Python packages dedicated to convex optimization ([20]). The simulated robot is a rather accurate model of the iCub robot which is present at ISIR in a version that does not allow torque control ([21]). 38 degrees of freedom are simulated (32 + 6 floating joints to locate the root in space), and 4 contact points at each foot, as shown in Figure 1.

A. Impedance Controller

As recalled in section II, impedance control provides a mean to handle the interaction with a changing environment: adapt to disturbances (e.g. an impact) or anticipate reactions (e.g. to push, grab, etc.).

In this simulation, iCub stands in front of a moving table which prints vertical oscillating motion with a period of 3 s.

[2]Of course, this rather simple supervisor can be coupled to or replaced by an higher level reasoning unit but this is out of scope of this work.

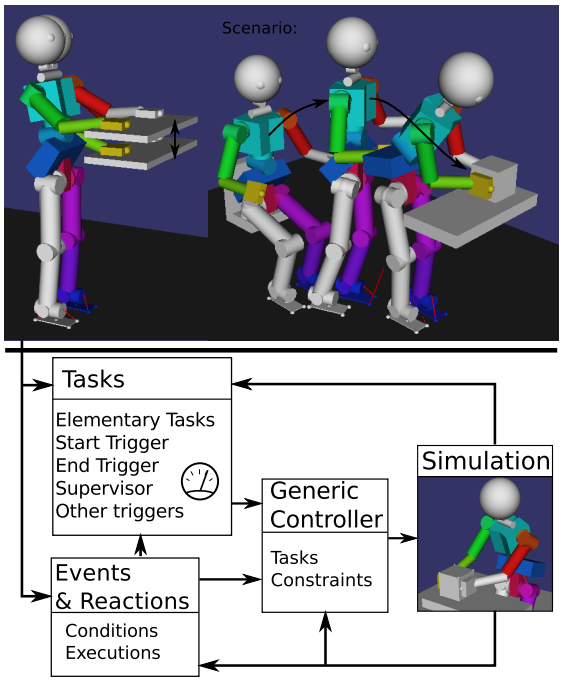

Fig. 2. Transformation of the scenario into tasks and triggering events which feed the generic controller.

The robot puts its hands on the table and applies a pressure using an impedance controller. Each hand has a reaching goal 10 cm below the point of contact, and a virtual spring-damper system linking the hand to its goal maintain the pressure on the table (stiffness and damping are respectively set to 10 s^{-2} and $2.\sqrt{10}$ s^{-1}). The robot keeps the pressure with its hands and the torques induced by this simulation have no discontinuity as shown on Figure 3.

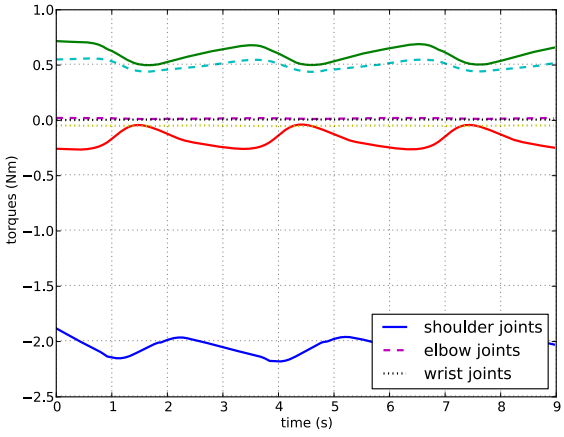

Fig. 3. An iCub robot applies a pressure on a moving table using an impedance controller.

B. Sequence of tasks

A complex scenario composed of a sequence of various complex tasks has been developed. This scenario describes iCub standing up from a chair, walking toward a table, grabbing a box and moving it to an other position, as shown on Figure 2. Using their respective weights, consecutive tasks are blended during a short period of time in order to ensure smooth transitions. For example, the walking task starts just before the end of the standing up task, and the same happens with the grabbing task. The mission is globally achieved with a good performance level (cf. the video submission attached to this paper) and rather than describing the entire mission in details, we hereafter focus on specific issues.

1) Hierarchy vs Weighting: In order to compare the weighting and hierarchy priority rules, the grabbing part of the scenario is performed using both strategies. In both cases four tasks (COM, hands, back, posture control) are used and associated either to strict priority values or to weights. Given the chosen tasks, the robot needs to bend over to achieve the box grabbing and this requires a modification of the relative tasks importances. This change is illustrated on Figure 4. The resulting input torso torques are shown on Figures 5 and 6. This result illustrates the positive impact of using smooth transitions in tasks priorities on the continuity of the control input.

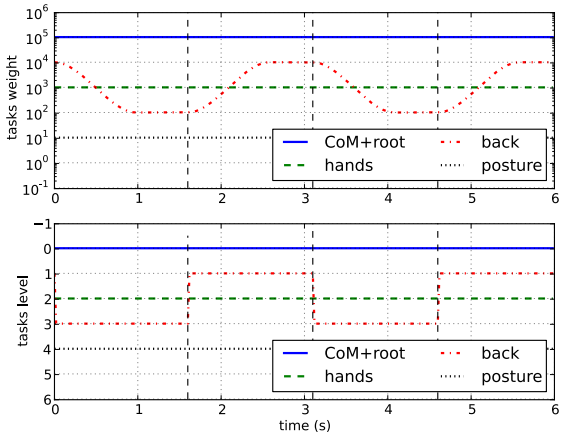

Fig. 4. Evolution of the tasks importance while grabbing the box.

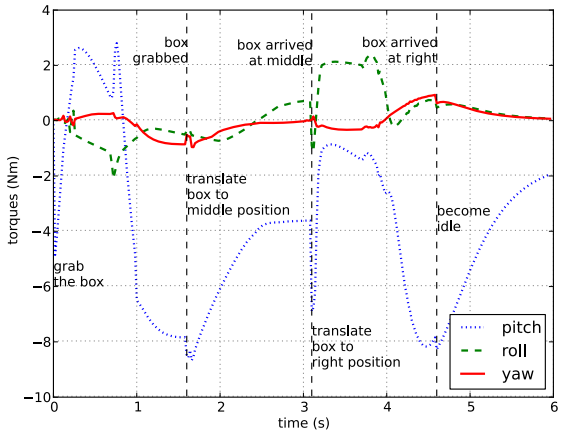

Fig. 5. Evolution of the input torques with a hierarchy strategy.

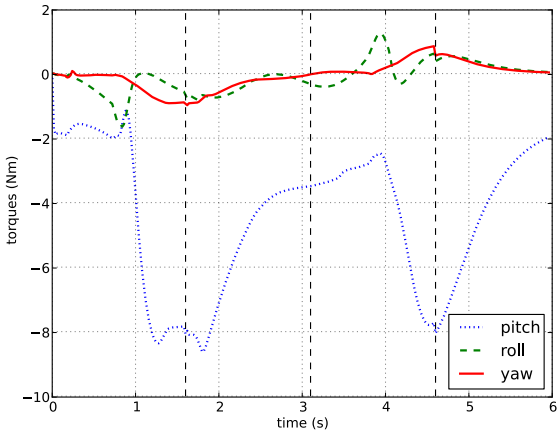

Fig. 6. Evolution of the input torques with a weighting strategy.

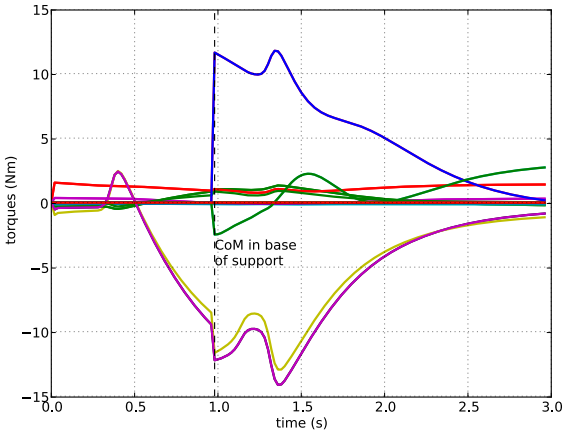

Fig. 7. Inputs torques applied to the robot without a constraint breaking strategy: a torque peak appears when contact vanishes.

2) Transition between constrained and free motions: We focus here on the sequencing of tasks inducing a transition between a constraint and a non-constraint state. The retained example is the standing behavior which leads to the disappearance of constraints when contacts vanish. A standing strategy involving a transition task is compared to a strategy where no transition is planned.

The input torques of the robot with no transition between constraint and non-constraint states is shown on Figure 7. The discontinuities are very important, and such a variation cannot be handled with a real system.

To cope with this issue, the constraints of the considered contacts are transformed into a transition task and then smoothly removed. In other words, if a contact disappears, a force control task is created with low but increasing priority over a short period of time, until the force becomes null. At this time, the constraint on the contact point acceleration must be replaced by an acceleration control task with a high but smoothly decreasing importance over a short period of time. If a contact appears, this set of operations is applied backward. The resulting input control torque is shown on Figure 8. Here again, the smooth insertion of transition tasks ensure smooth control inputs.

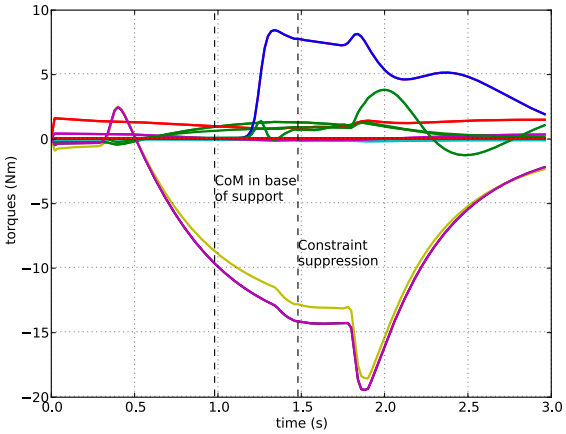

Fig. 8. Inputs torques applied to the robot with a constraint breaking strategy: the control torque is increased as expected but in a smooth manner.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper, we present a method to perform a sequence of dynamic tasks. We use an optimization program to solve the problem of computing the proper torque control given a set of tasks subject to a set of equality and inequality constraints.

The main contribution of this paper is the development of a strategy to define soft priorities between tasks that have to be achieved simultaneously. The evolution of the tasks importance is ensured with weights and allows to easily perform smooth sequencing between changing tasks and, as a consequence, to synthesize complex behaviors.

This weighting strategy as well as the insertion of transitional tasks also endow the system with the capability to switch smoothly between constrained and non-constrained states and ensure that continuous input torques are fed to the robot.

Simulation results on a virtual iCub robot are presented to illustrate the realization of a complex mission as well as some specific aspects related to tasks transitions.

B. Future Works

A simple extension to this work will consist in melding both weighting and strict hierarchy strategies. Indeed, even though the soft priority strategy is shown to be more versatile, exhibits good properties from the control standpoint and is computationally less expensive, one can imagine cases where very specific tasks cannot be handled as constraints but needs to be achieved almost perfectly (e.g. balance tasks on complex grounds).

Also, the proposed framework was not tested in the case of collision avoidance tasks. We are confident that this type of tasks can be implemented in a straightforward manner and this will be part of near future work.

From the supervision point of view, automatic tasks weighting and transition insertions can probably be improved

in order to rely on a less *had oc* strategy and provide a more general framework.

Finally, a new version of iCub allowing torque control will soon be available at ISIR and will allow us to implement and validate this control algorithm on a real robot.

REFERENCES

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, 2003.

[2] D. Dimitrov, P.-B. Wieber, O. Stasse, J. Ferreau, and H. Diedam, "An optimized linear model predictive control solver for online walking motion generation," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Kobe, Japan, 2009.

[3] J. Park and O. Khatib, "Robot multiple contact control," *Robotica*, vol. 26, no. 5, pp. 667–677, 2008.

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

[5] L. Sentis, J. Park, and O. Khatib, "Compliant control of multi-contact and center of mass behaviors in humanoid robots," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 483–501, June 2010.

[6] Y. Abe, M. da Silva, and J. Popovic, "Multiobjective control with frictional contacts," in *Proceedings of the Symposium on Computer Animation*, 2007.

[7] C. Collette, A. Micaelli, P. Lemerle, and C. Andriot, "Robust balance optimization control of humanoid robots with multiple non coplanar grasps and frictional contacts," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Pasadena, USA, May 2008, pp. 3187–3193.

[8] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Anchorage, USA, May 2010.

[9] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter, "Extending itasc to support inequality constraints and non instantaneous task specification," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Kobe, Japan, May 2009.

[10] J. Salini, S. Barthélémy, and P. Bidaud, *Advances in Robot Kinematics: Motion in Man and Machine*. Springer, 2010, ch. LQP-Based Controller Design for Humanoid Whole-Body Motion, pp. 177–184.

[11] V. Duindam, "Port-based modeling and control for efficient bipedal walking robots," Ph.D. dissertation, University of Twente, 2006.

[12] S. L. Altmann, *Rotations, Quaternions and double groups*. Oxford University Press, 1986.

[13] N. Hogan, "Stable execution of contact tasks using impedance control," in *Proceedings of the IEEE International Conference on Robotics & Automation*, mar 1987, pp. 1047–1054.

[14] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Genova, Italy, 2006.

[15] J. Park, J. Han, and F. C. Park, "Convex optimization algorithms for active balancing of humanoid robots," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 817–822, 2007.

[16] S. Rubrecht, V. Padois, P. Bidaud, and M. De Broissia, "Constraints Compliant Control: constraints compatibility and the displaced configuration approach," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010.

[17] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond, "Prioritizing linear equality and inequality systems: Application to local motion planning for redundant robots," in *Proceedings of the IEEE International Conference on Robotics & Automation*, Kobe, Japan, May 2009, pp. 2939–2944.

[18] N. Mansard, A. Remazeilles, and F. Chaumette, "Continuity of varying-feature-set control laws," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, November 2009.

[19] S. Barthélemy and A. Micaelli, "Arboris for Matlab." [Online]. Available: http://vizir.robot.jussieu.fr/trac/arboris

[20] J. Dahl and L. Vandenberghe, "cvxopt - python software for convex optimization." [Online]. Available: http://abel.ee.ucla.edu/cvxopt/

[21] G. Sandini, G. Metta, and D. Vernon, "The icub cognitive humanoid robot: An open-system research platform for enactive cognition," in *50 Years of Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer, 2007, ch. 32, pp. 358–369.