2013 13th IEEE-RAS International Conference on
Humanoid Robots (Humanoids).
October 15 - 17, 2013. Atlanta, GA

Real-Time Smooth Task Transitions for Hierarchical Inverse Kinematics

Gerardo Jarquín, Adrien Escande, Gustavo Arechavaleta, Thomas Moulard,
Eiichi Yoshida and Vicente Parra-Vega

***Abstract*— Hierarchical inverse kinematics (HIK) is widely used for generating feasible velocity trajectories that serve as input references for highly redundant robots such as humanoid robots. To generate the velocity trajectories a set of prioritized tasks should be provided. For some applications, it is not necessary to change the priority order of the tasks in the stack of tasks (SoT) along the motion execution. However, complex tasks need a dynamic behavior of the SoT such that the insertion, removal or swap can be performed at running time. These task transitions may induce discontinuities in the joint velocities if they are not carefully handled. In this context, we propose an efficient strategy to manage task transitions through a simple procedure which smoothly interchange the priority of a couple of consecutive prioritized tasks. Furthermore, our method does not increase the computational cost of the HIK since neither any additional task should be added, nor parallel control laws should be computed. As a result our strategy may be used in real time to produce the velocity commands of real humanoid robots. The effectiveness of our strategy is verified at simulation level with the HRP-2 humanoid robot performing complex time-driven tasks.**

I. Introduction

For the last two decades, humanoid robots have gained special attention in the robotics community due to the challenges that their highly redundant mechanical structure represent. Moreover, its motion capabilities make them suitable to perform tasks in human environments [1], [2].

In the context of whole-body motion generation, the task priority approach [3] is an effective tool to exploit the redundant structure of humanoid robots [4]. Dynamic and kinematic controllers have been proposed to simultaneously handle a set of prioritized constraints [5], [6], [7], also called tasks. Then, the set of tasks is ordered in a structure called stack of tasks (SoT) [8].

Commonly, the SoT has a static behavior, i.e. the tasks and their priorities do not change along the motion execution. The tasks are defined once at the beginning and remain unchanged until the motion terminates. For solving more complex motion problems, a dynamic SoT is required such that any task is subject to be added, removed or its current priority can be changed. These actions are called task transitions and it is well known that they may induce discontinuities in the joint velocities [9]. Recently, the work in [10] proposed a strategy called intermediate desired value approach (IDVA). The underlying idea relies on the modification of the desired end-effector velocities of the task in transition by adding a term which represents the velocity of the end-effector produced by the other tasks in the SoT. This new term is called intermediate desired value. Then the smooth transition between this term and the original desired velocity allows the insertion, removal or swap of tasks. Despite the effectiveness of this method, the computational cost increases exponentially with respect to the number of tasks in transition since the HIK is computed for each intermediate desired value. This represents a drawback for real-time implementation purposes. Another solution to avoid discontinuities due to task transitions is provided in [9]. The solution consists in applying a linear interpolation between two control laws. This strategy is less costly than the IDVA since it is only necessary to partially interpolate the two control laws. In one way or another, the computational cost increases because its implementation requires to modify the HIK solver in order to doubly compute the tasks in transition.

G. Jarquín, G. Arechavaleta and V. Parra-Vega are with the Robotics and Advanced Manufacturing Group, Centro de investigacin y de Estudios Avanzados del IPN, Saltillo, Coah. México {gerardojarquin, garechav, vicente.parra}@cinvestav.edu.mx.

A. Escande, T. Moulard and E. Yoshida are with the CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/CRT {adrien.escande, thomas.moulard, e.yoshida}@aist.go.jp

A. Problem statement and contribution

The present work aims at generating smooth trajectories for humanoid robots under task transitions. The inputs are the set of prioritized tasks, the desired SoT after each transition and the time when transitions should occur. As a result, the method should be capable to produce feasible velocity trajectories to comply with input requirements.

The contribution of the paper is then to provide an efficient method for swapping the priorities of two consecutive tasks. It takes advantage of a weighting method to merge in the same priority two tasks coming from two consecutive priority levels. In order to comply with the original priorities, proper weights are assigned to each task. Such weights are smoothly changed to modify the tasks priorities such that the original priorities are inverted. It is worth mentioning that our method does not modify the inner computation of HIK solvers since it only acts at the definition of the tasks. Consequently, any available HIK solver in the literature can use the proposed method.

The remaining of the paper is structured as follows. Section II briefly recalls the task priority formalism. In section III is discussed the problem of task transitions and the most recent schemes in the literature offering a solution. Then, the main result is presented in Section IV. The simulation results that validate our method are described in Section V. Finally, we provide some final remarks in Section VI.

978-1-4799-2618-3/13/$31.00 ©2013 IEEE

II. TASK PRIORITY FORMALISM

The priority of each task in the SoT ensures that tasks with lower priority do not perturb the higher priority ones. This behavior is achieved by solving each task within the null space of all tasks with higher priority. In the remaining of this section we recall the kinematic version of this formalism.

Let us consider an equality task that may be characterized by a differential function of the form

$$\dot{\boldsymbol{x}}_1 - \dot{\boldsymbol{x}}_{d_1} = 0 \tag{1}$$

where $\dot{\boldsymbol{x}}_{d_1}, \dot{\boldsymbol{x}}_1 \in \mathbb{R}^m$ are the desired and current velocity vectors of the end-effector 1, respectively. The first term on the left is a function of the desired and actual end-effector pose vectors $\dot{\boldsymbol{x}}_1 = f(\boldsymbol{x}_1, \boldsymbol{x}_{d_1})$. It can be represented as a time varying function in order to drive the pose error $\boldsymbol{e}_1 = \boldsymbol{x}_1 - \boldsymbol{x}_{d_1}$ to zero (see [11], [8], [12] and [5]). The second term on the left is obtained from the robot state through the differential kinematic equation

$$\dot{\boldsymbol{x}}_1 = J_1(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2}$$

where $J_1(\boldsymbol{q}) \in \mathbb{R}^{m\times n}$ is the Jacobian matrix, $\boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^n$ are the vectors of joint positions and joint velocities, respectively. Note that for the case of redundant robots $n > m$.

By substituting (2) in (1) the task function is obtained

$$J_1(\boldsymbol{q})\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}_{d_1} = 0 \tag{3}$$

which solution for $\dot{\boldsymbol{q}}$, in the least squares sense, is [3]

$$\dot{\boldsymbol{q}} = J_1(\boldsymbol{q})^+ \dot{\boldsymbol{x}}_{d_1} + Q_1 \dot{\boldsymbol{q}}_2 \tag{4}$$

where the operator $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse. The second term represents the residual space remaining after the solution of the task 1. Thus, Q_1 is a projector onto the null space of $J_1(\boldsymbol{q})$

$$Q_1 = I_n - J_1(\boldsymbol{q})^+ J_1(\boldsymbol{q}) \tag{5}$$

with $I_n \in \mathbb{R}^{n\times n}$ as the identity matrix. Note that $\dot{\boldsymbol{q}}_2$ may be the solution of another task function $J_2(\boldsymbol{q})\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}_{d_2} = 0$. The solution for both tasks is

$$\dot{\boldsymbol{q}}_2 = J_1(\boldsymbol{q})^+ \dot{\boldsymbol{x}}_{d_1} + (J_2(\boldsymbol{q})Q_1)^+ \left(\dot{\boldsymbol{x}}_{d_2} - J_2(\boldsymbol{q})J_1(\boldsymbol{q})^+ \dot{\boldsymbol{x}}_{d_1}\right) \tag{6}$$

where $(J_2(\boldsymbol{q})Q_1)^+$ is the projector to the range space of the projection of $J_2(\boldsymbol{q})$ in the null space of $J_1(\boldsymbol{q})$. Since the second task is solved in the null space of the first task, then this implies that task 1 has higher priority than task 2. This process can be recursively applied to solve as many tasks as available degrees of freedom (DoF) has the robot. The recursive formulation for $k = 2$ to p tasks becomes [13]:

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + (J_k(\boldsymbol{q})Q_{k-1})^+ \left(\dot{\boldsymbol{x}}_{d_k} - J_k(\boldsymbol{q})\dot{\boldsymbol{q}}_{k-1}\right) \tag{7}$$

with $\dot{\boldsymbol{q}}_1 = J_1(\boldsymbol{q})^+ \dot{\boldsymbol{x}}_{d_1}$ and Q_1 given by Eq. (5), respectively. The projector Q_k can be efficiently computed in linear time as in [14]:

$$Q_k = Q_{k-1} - (J_k(\boldsymbol{q})Q_{k-1})^+ J_k(\boldsymbol{q})Q_{k-1}$$

Note that the recursive solution (7) can be also obtained from the following constrained quadratic program [15]:

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}_k, \boldsymbol{w}_k} \ & \frac{1}{2} ||\boldsymbol{w}_k||^2 \\ s.t. \quad & J_k(\boldsymbol{q})\dot{\boldsymbol{q}}_k - \dot{\boldsymbol{x}}_{d_k} = \boldsymbol{w}_k \\ & \bar{J}_{k-1}(\boldsymbol{q})\dot{\boldsymbol{q}}_k - \bar{\dot{\boldsymbol{x}}}_{d_{k-1}} = \bar{\boldsymbol{w}}^*_{k-1} \end{aligned} \tag{8}$$

where $\boldsymbol{w}_k \in \mathbb{R}^m$ is a vector of slack variables used to relax the infeasible constraints in the priority level k, $\boldsymbol{w}^*_k$ represents the optimum value of $\boldsymbol{w}_k$. The bar above some variables means a stack, such that $\bar{A}_k = \begin{bmatrix} A_1^T & A_2^T & \dots & A_k^T \end{bmatrix}^T$.

III. THE TASK TRANSITION PROBLEM

A task transition occurs when the order of the tasks in the SoT changes. This change may be produced by the insertion or removal of at least one task, but also it may be produced because at least two tasks interchange their respective priority levels. The simplest case implies the interchange of a pair of consecutive tasks according to the hierarchy. The problem with changing the order of the tasks in the SoT lies on the fact that two different order of tasks produce different joint velocities in the general case. Then, when a SoT is abruptly changed by another, a discontinuity is produced. In the remaining of this section, two representative schemes capable to avoid such discontinuities are discussed.

A. Linear interpolation

A direct solution to the task transition problem is to pass smoothly from the solution of a SoT with a certain order of tasks to another SoT by performing a linear interpolation between both solutions [9]. The strategy applies a number of swapping operations with consecutive tasks. For instance, the insertion of a task means that it is activated in the null space of all the active tasks, then through several swaps it is brought to the desired priority level. The reverse process is performed to remove the task.

Let us consider two SoT, SoT_{AB} and SoT_{BA}. Both are composed by the same tasks with consecutive priority levels, i.e. A and B. In the SoT_{AB}, the task A has higher priority than task B while SoT_{BA} represents the opposite hierarchy. Let i be the task with the next higher priority level than tasks A and B, similarly, let j be the task with the next lower priority level. Since the order of tasks with higher priority level than tasks A and B is the same in both SoT, the solution for the upper levels is the same. Such a solution is computed from (7) with $k = 2, \dots, i$. However, for levels A and B the solution is different according to the priority order. For the SoT_{AB}

$$\begin{aligned} \dot{\boldsymbol{q}}_A &= \dot{\boldsymbol{q}}_i + (J_A(\boldsymbol{q})Q_i)^+ (\dot{\boldsymbol{x}}_{d_A} - J_A(\boldsymbol{q})\dot{\boldsymbol{q}}_i) \\ \dot{\boldsymbol{q}}_{AB} &= \dot{\boldsymbol{q}}_A + (J_B(\boldsymbol{q})Q_A)^+ (\dot{\boldsymbol{x}}_{d_B} - J_B(\boldsymbol{q})\dot{\boldsymbol{q}}_A) \end{aligned} \tag{9}$$

where $Q_A = Q_i - (J_A(\boldsymbol{q})Q_i)^+ J_A(\boldsymbol{q})Q_i$. On the other hand, for the case of SoT_{BA} we have:

$$\begin{aligned} \dot{\boldsymbol{q}}_B &= \dot{\boldsymbol{q}}_i + (J_B(\boldsymbol{q})Q_i)^+ (\dot{\boldsymbol{x}}_{d_B} - J_B(\boldsymbol{q})\dot{\boldsymbol{q}}_i) \\ \dot{\boldsymbol{q}}_{BA} &= \dot{\boldsymbol{q}}_B + (J_A(\boldsymbol{q})Q_B)^+ (\dot{\boldsymbol{x}}_{d_A} - J_A(\boldsymbol{q})\dot{\boldsymbol{q}}_B) \end{aligned} \tag{10}$$

with $Q_B = Q_i - (J_B(\boldsymbol{q})Q_i)^+ J_B(\boldsymbol{q})Q_i$. Then, the priority level j is given by

$$\dot{\boldsymbol{q}}_j = \dot{\boldsymbol{q}}_i + \hat{\dot{\boldsymbol{q}}} + \left(J_j(\boldsymbol{q})\hat{Q}\right)^+ \left(\dot{\boldsymbol{x}}_{d_j} - J_j(\boldsymbol{q})\hat{\dot{\boldsymbol{q}}}\right)$$

where

$$\hat{\dot{\boldsymbol{q}}} = \begin{cases} \dot{\boldsymbol{q}}_{AB} & \text{for} \quad SoT_{AB} \\ \dot{\boldsymbol{q}}_{BA} & \text{for} \quad SoT_{BA} \end{cases} \tag{11}$$

From the fact that the projector Q_k represents the null space of all the k tasks, it is easy to deduce that

$$\begin{aligned} \hat{Q} &= Q_A - (J_B(\boldsymbol{q})Q_A)^+ J_B(\boldsymbol{q})Q_A \\ &= Q_B - (J_A(\boldsymbol{q})Q_B)^+ J_A(\boldsymbol{q})Q_B \end{aligned}$$

then the computation of the last priority levels is not affected by the order of A and B. These levels are computed from (7) with $k = j+1, \ldots, p$.

Consequently, to change the priority order of tasks A and B it is only necessary to perform a linear interpolation between $\dot{\boldsymbol{q}}_{AB}$ and $\dot{\boldsymbol{q}}_{BA}$

$$\hat{\dot{\boldsymbol{q}}} = \xi\dot{\boldsymbol{q}}_{AB} + (1-\xi)\dot{\boldsymbol{q}}_{BA} \tag{12}$$

where ξ is a time varying function that smoothly evolves from 0 to 1.

Despite this solution is simple and easy to implement, it has two main inconvenients. First, the computational cost increases since the partial solutions (9) and (10) should be computed at the same time during the transition period. This characteristic adds the computation of two pseudoinversions for each swap operation. Then it is necessary to compute $p+2n_s$ pseudoinversions for n_s number of swaps performed at the same time. Note that n_s implies 2 tasks in transition. It is clear that for $n_s < p/2$, the number of pseudoinversions are equivalent to call the HIK solver less than 2 but more than 1 times. The worst case is when all tasks are in transition, i.e. $n_s = p/2$. In this case the computational cost is equivalent to call twice the HIK solver. Second, it needs to modify the HIK solver in order to allow the double computation of the levels in transition. This may be an obstacle to exploit the nice characteristics of recent HIK solvers such as [15], which compute the control law in one operation after a matrix decomposition process based on QR factorizations. As a consequence, in order to not modify the HIK solver, it would be necessary to interpolate two complete control laws.

B. Intermediate desired value approach

An alternative and effective solution to smoothly manage task transitions is proposed in [10], called intermediate desired value approach (IDVA). With this scheme it is possible to insert or remove a task in one step. The main idea relies on the modification of the desired end-effector velocity in order to change smoothly from the velocity of the end-effector produced by the other active tasks to the desired velocity.

Let i be a task that should be removed or inserted, within the transition period the desired end-effector velocity changes as follows

$$\hat{\dot{\boldsymbol{x}}} = \xi\dot{\boldsymbol{x}}_{d_i} + (1-\xi)J_i\dot{\boldsymbol{q}}_{p/i} \tag{13}$$

where $\hat{\dot{\boldsymbol{x}}}$ is the intermediate desired value for task i, $\dot{\boldsymbol{q}}_{p/i}$ is the solution of the p tasks in the SoT without considering the task i. In (13), it is clear that if $\xi = 0$ the desired end-effector velocity is actually the current velocity produced by the action of the other active tasks. Thus, by changing ξ the behavior of the end effector changes from the action of the other tasks to the desired velocity.

This strategy effectively allows the smooth insertion, removal and swap of any number of tasks without modifying the HIK solver since it acts only on the desired end-effector velocity. However it has the inconvenient that, in the general case, the HIK solver function should be called $2^{n_{tr}} - 1$ times for n_{tr} tasks in transition, i.e. the computational cost increases exponentially accordingly with the number of tasks in transition. This behavior is a consequence of the computation of $\dot{\boldsymbol{q}}_{p/i}$, which depends on the computation of the other intermediate desired values.

IV. Smooth priority swap strategy

In this section we present a new strategy that is suitable to be applied in real time since it does not increase the computational cost of the HIK solver. Moreover, this strategy is independent to the HIK solver since it acts at the definition of each task in the SoT. The strategy is based on the results presented in [16] which allow us to define a transition phase where the two tasks involved in the swap are merged. Then the weights of each task are smoothly changed in order to invert the original priorities.

A. Priority swap of two consecutive tasks

The objective is to design a transition phase in the period $t_{tr}^0 \leq t \leq t_{tr}^f$ (see Fig. 1) which helps to change smoothly from the solution given by the minimization problem (14) to the one given by (15)

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}_k, \boldsymbol{w}_k} \frac{1}{2}\,||\boldsymbol{w}_k||^2 \\ s.t. \quad &{}^B J_k(\boldsymbol{q})\dot{\boldsymbol{q}}_k - {}^B\dot{\boldsymbol{x}}_{d_k} = {}^B\boldsymbol{w}_k \\ &{}^A J_{k-1}(\boldsymbol{q})\dot{\boldsymbol{q}}_k - {}^A\dot{\boldsymbol{x}}_{d_{k-1}} = {}^A\boldsymbol{w}^*_{k-1} \\ &\bar{J}_{k-2}(\boldsymbol{q})\dot{\boldsymbol{q}}_k - \bar{\dot{\boldsymbol{x}}}_{d_{k-2}} = \bar{\boldsymbol{w}}^*_{k-2} \end{aligned} \tag{14}$$

where the upper-scripts A and B are used to identify the tasks before and after the swap operation such that before the swap, task A has higher priority than task B.

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}_k, \boldsymbol{w}_k} \frac{1}{2}\,||\boldsymbol{w}_k||^2 \\ s.t. \quad &{}^A J_k(\boldsymbol{q})\dot{\boldsymbol{q}}_k - {}^A\dot{\boldsymbol{x}}_{d_k} = {}^A\boldsymbol{w}_k \\ &{}^B J_{k-1}(\boldsymbol{q})\dot{\boldsymbol{q}}_k - {}^B\dot{\boldsymbol{x}}_{d_{k-1}} = {}^B\boldsymbol{w}^*_{k-1} \\ &\bar{J}_{k-2}(\boldsymbol{q})\dot{\boldsymbol{q}}_k - \bar{\dot{\boldsymbol{x}}}_{d_{k-2}} = \bar{\boldsymbol{w}}^*_{k-2} \end{aligned} \tag{15}$$

According to [16], it is possible to change the constrained minimization problem

$$\begin{aligned} &\min_{\mathrm{x}} \frac{1}{2}\,\|G\mathrm{x} - g\|^2 \\ s.t. \quad &H\mathrm{x} - h = 0 \end{aligned} \tag{16}$$

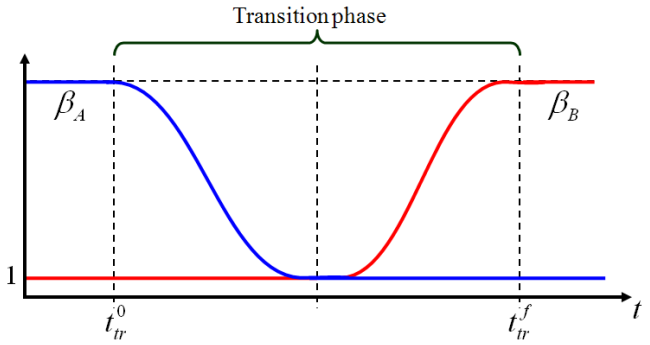

Fig. 1. Profiles of the weights during the transition phase. The decreasing weight β_A leads the task with initial higher priority level to lose its priority when $\beta_A = 1$. Then, the weight β_B smoothly provides with higher priority to task B.

for an unconstrained one of the form

$$\min_{\dot{\boldsymbol{q}}} \frac{1}{2} \left\| \begin{bmatrix} G \\ \beta H \end{bmatrix} \mathrm{x} - \begin{bmatrix} g \\ \beta h \end{bmatrix} \right\|^2 \tag{17}$$

where β establishes the weight of the task $Hx - h$ over the task $Gx - h$. Consequently, with an adequate value for β it is possible to set a priority for the tasks in a similar way than (16). Based on this result, we propose to design the transition phase by merging both priority levels in transition, in order to solve the constrained quadratic program

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}_k, {}^A\boldsymbol{w}_k, {}^B\boldsymbol{w}_{k-1}} \quad & \frac{1}{2} \left\| \begin{bmatrix} \beta_A(t)({}^A\boldsymbol{w}_k) \\ \beta_B(t)({}^B\boldsymbol{w}_{k-1}) \end{bmatrix} \right\|^2 \\ s.t. \quad {}^A J_k(\boldsymbol{q})\dot{\boldsymbol{q}}_k - {}^A\dot{\boldsymbol{x}}_{d_k} &= {}^A\boldsymbol{w}_k \\ {}^B J_{k-1}(\boldsymbol{q})\dot{\boldsymbol{q}}_k - {}^B\dot{\boldsymbol{x}}_{d_{k-1}} &= {}^B\boldsymbol{w}_{k-1} \\ \bar{J}_{k-2}(\boldsymbol{q})\dot{\boldsymbol{q}}_k - \bar{\dot{\boldsymbol{x}}}_{d_{k-2}} &= \bar{\boldsymbol{w}}^*_{k-2} \end{aligned} \tag{18}$$

The idea is to switch from (14) to (18) at $t = t^0_{tr}$ with $\beta_B(t^0_{tr}) = 1$. The value of $\beta_A(t^0_{tr})$ should be selected to preserve the priority of task A over the task B such that the solutions of (14) and (18) are equivalent. As a consequence, the continuity is ensured at the switching instant. Then, $\beta_A(t)$ should decrease its value until reach 1, at that moment both tasks have the same value and, consequently, the same priority. Then $\beta_B(t)$ begins its variation toward its final value which will be reached at $t = t^f_{tr}$, at this time the minimization problem is switched to (15) and the swap process is concluded. The continuity of the solution at $t = t^f_{tr}$ depends on the value of $\beta_B(t^f_{tr})$ which should be selected in order to comply with the priority order of (15). Note that $\beta_A(t^0_{tr}), \beta_B(t^f_{tr}) \geq 1$ and since there is only two tasks in the same priority level these values are not difficult to choose but they should be properly tunned.

It is important to mention that in [9] it is also discussed a strategy to swap the priorities of two consecutive tasks by changing the weights of two tasks temporally placed at the same level of hierarchy. Such a strategy was discarded due to the impossibility of using the classic damping method to avoid the algorithmic singularities. In order to establish the difference between our strategy and that proposed in [9], Section IV.A, let us recall with our notation the formulation for two tasks used in [9]

$$\min_{\dot{\boldsymbol{q}}} \frac{1}{2} \left\| \begin{bmatrix} (1-\alpha(t))({}^A J(\boldsymbol{q})\dot{\boldsymbol{q}} - {}^A\dot{\boldsymbol{x}}_d) \\ \alpha(t)({}^B J(\boldsymbol{q})\dot{\boldsymbol{q}} - {}^B\dot{\boldsymbol{x}}_d) \end{bmatrix} \right\|^2 + \delta\|\dot{\boldsymbol{q}}\|^2 \tag{19}$$

where $\lim_{t \to t^0_{tr}} \alpha(t) = 0$ and $\lim_{t \to t^f_{tr}} \alpha(t) = 1$. The variation of the weights of tasks A and B as a consequence of the evolution of $\alpha(t)$ produce a discontinuity when the weight of task A becomes lower than δ because task A is suddenly shadowed, i.e. the task A is abruptly deactivated because the space used for its regulation becomes empty. A similar case occurs when the weight of task B becomes higher than δ. In contrast with this strategy, in our method the weights of tasks A and B never are lower than 1, this means that the damping is always the task with the third priority level. Consequently, it is possible to use the classic damping method to prevent high joint velocities produced by algorithmic singularities.

B. Insertion and removal of tasks

The strategy to insert and remove a task is based on sequential swaps. In order to insert a task, first it should be inserted in the last priority level by merging the new task with the task with lowest priority, the problem to solve in the transition phase is

$$\begin{aligned} \min_{\dot{\boldsymbol{q}}_k, {}^A\boldsymbol{w}_p, {}^B\boldsymbol{w}_{p+1}} \quad & \frac{1}{2} \left\| \begin{bmatrix} {}^A\boldsymbol{w}_p \\ \beta_{ins}(t)({}^B\boldsymbol{w}_{p+1}) \end{bmatrix} \right\|^2 \\ s.t. \quad {}^A J_{p+1}(\boldsymbol{q})\dot{\boldsymbol{q}}_{p+1} - {}^A\dot{\boldsymbol{x}}_{d_{p+1}} &= {}^A\boldsymbol{w}_{p+1} \\ {}^B J_p(\boldsymbol{q})\dot{\boldsymbol{q}}_{p+1} - {}^B\dot{\boldsymbol{x}}_{d_p} &= {}^B\boldsymbol{w}_p \\ \bar{J}_{p-1}(\boldsymbol{q})\dot{\boldsymbol{q}}_{p+1} - \bar{\dot{\boldsymbol{x}}}_{d_{p-1}} &= \bar{\boldsymbol{w}}^*_{p-1} \end{aligned} \tag{20}$$

however in this case $\beta_{ins}(t^0_{tr}) = 0$ and it increases to its final value at $t = t^f_{tr}$, this variation is different due to the fact that the task $p+1$ was not in the SoT. In order to bring the task to the desired priority level, a set of swaps by means the transition phase (18) must be performed. It is important to note that after one transition phase it is possible to switch to another transition phase in the next priority level, thus, only when the task is in the desired priority level it will be necessary to switch to (14) such that the time spent to bring the task to its final priority level is as low as possible. It is worth noticing that since $\beta_{ins}(t^0_{tr}) = 0$ a discontinuity is produced when a damping factor is used as explained above. However, this problem can be solved by smoothly decreasing the value of δ until zero before the transition phase (20).

To implement our method it is only necessary to define the proper values of $\beta_A(t^0_{tr})$ and $\beta_B(t^f_{tr})$ in order to comply with the equivalence of the solution given by (18) and the solutions provided by (14) and (15). It is important to mention that the insertion needs to spend some time to bring the task from the last priority to the desired one since it is necessary to perform several swaps. The same applies to remove a task. This may produce a relative slow response of the robot in applications where the robot should react quickly by removing or adding a task. It is evident that by reducing the transition period the time spent to place a task in its

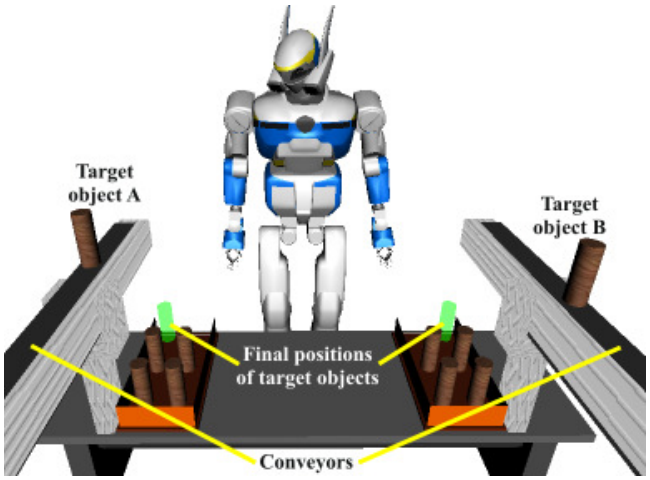

Fig. 2. Simulation scenario. The HRP-2 robot must grasp two mobile wooden objects in order to put them in their respective boxes. As the conveyor brings the objects to the robot, it must grasp them before they reach the edge of the conveyor.

desired priority level also decreases, however this action may produce fast joint velocity variations, although continuous.

V. Simulation Results

In order to validate our strategy we design the simulation scenario shown in Fig 2. The scenario consists mainly of two conveyors transporting cylindrical wooden objects, a humanoid robot HRP-2 and a couple of boxes containing some wooden objects. The objective is that the HRP-2 grasps the objects approaching on the conveyors before they reach the edge, otherwise, they will fall. After taking the objects, the robot has to put them into the boxes on the table. Since we are considering tasks driven by time, we used the time base generator gain presented in [5] to manage the completion time of the reaching tasks.

The simulation begins with the objects on the conveyors approaching the HRP-2, the object on the right hand is the nearest from the robot and the first one to be reached. In table I it is shown the order of the tasks in the SoT at some critical time instants. Note that at the beginning of the simulation, the task of the right hand (RHT) is already active while the task of the left hand (LHT) is not. After taking the object on the right, the robot has to put it in the right box, however the object on the left is near the conveyor edge, as a consequence, the LHT needs to be activated but also it needs to have a higher priority than the RHT. Because it is not possible that the robot can reach the target in the right box and the object in the left conveyor at the same time, the higher priority of the LHT will allow the robot to reach the object in the conveyor even when the target in the box cannot be completed. However, in order to provide with more DoF to the LHT, the RHT could be removed. As a consequence the robot reaches the left object with a better posture. Once both objects have been reached, the robot has to put them into the boxes. Since a new object in the right conveyor is approaching the robot, it should have available the right hand to take the new object, then the RHT is activated and provided with higher priority than the LHT. In order to generate a better posture for the robot, it is a better choice to remove the LHT while the robot reach the right target in the box. After the robot places the object in the desired position, the RHT is removed and LHT is activated to place the left object inside the left box. Both robot hands are now free and a new cycle may begin. A series of snapshots of the motion execution is shown in the first row of Fig. 3 where it can be observed that the DoF released when a task is removed are occupied to maintain the robot balance.

The simulation results for the joint velocities are shown in the second row of Fig (3). Note that the velocity profiles are smooth along the simulation even when the tasks are removed or inserted.

VI. Conclusion and Future Work

We presented an effective method to smoothly manage task transitions of a set of prioritized tasks based on a weighting strategy. Our method is appropriate for being used in real time since it does not require to modify the HIK solver. Consequently, the computational cost depends only on the selection of the HIK solver used to generate the velocity commands. The strategy is based on a mechanism to smoothly swap the priority of two consecutive tasks. The insertion of a task is achieved by performing a sequence of swaps to bring the new task from the last priority level to the desired one. In order to remove a task the inverse process is followed. This process requires to spend some time to insert or remove a task, which may be an inconvenient in the strategy when the application requires that the insertion or remotion should be carried out quickly. We are working on extending our strategy to use it together with HIK solvers capable to handle inequality constraints in real time as the presented in [15]. The natural next step is to consider dynamic constraints by integrating our method in prioritized inverse dynamics schemes.

References

[1] H. Harada, S. Kajita, H. Saito, M. Morisawa, F. Kanehiro, K. Fujiwara, K. Kaneko, and H. Hirukawa, "A humanoid robot carrying a heavy object," in *IEEE International Conference on Robotics and Automation*, Barcelona spain, April 2005, pp. 1712–1717.

[2] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human enviroments," in *IEEE International Conference on Robotics and Automation*, Orlando, FL, USA, May 2006, pp. 2641–2648.

[3] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, no. 12, pp. 868–871, December 1977.

[4] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *IEEE/RAS International Conference on Humanoids Robots*, Tsukuba, Japan, December 2005, pp. 238–244.

[5] G. Jarquín, G. Arechavaleta, and V. Parra-Vega, "Continuous kinematic control with terminal attractors for handling task transitions of redundant robots," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.

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

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

TABLE I

COMPOSITION OF THE STACK OF TASKS AT SOME CRITIC TRANSITION TIMES

P	0 seconds	1.5 seconds	3 seconds	3.8 seconds	4.6 seconds	6 seconds	7 seconds
1	Feet poses						
2	Center of mass position						
3	Right hand pose	Right hand pose	Left hand pose	Left hand pose	Right hand pose	Right hand pose	Left hand pose
4	Waist orientation	Waist orientation	Waist orientation	Waist orientation	Waist orientation	Waist orientation	Waist orientation
5	Sight orientation	Sight orientation	Sight orientation	Sight orientation	Sight orientation	Sight orientation	Sight orientation
6	Chest pitch	Chest pitch	Chest pitch	Chest pitch	Chest pitch	Chest pitch	Chest pitch
7	Avoid obstacles	Left hand pose	Avoid obstacles	Right hand pose	Avoid obstacles	Left hand pose	Avoid obstacles
8		Avoid obstacles		Avoid obstacles		Avoid obstacles	
Idle	Left hand pose		Right hand pose		Left hand pose		Right hand pose

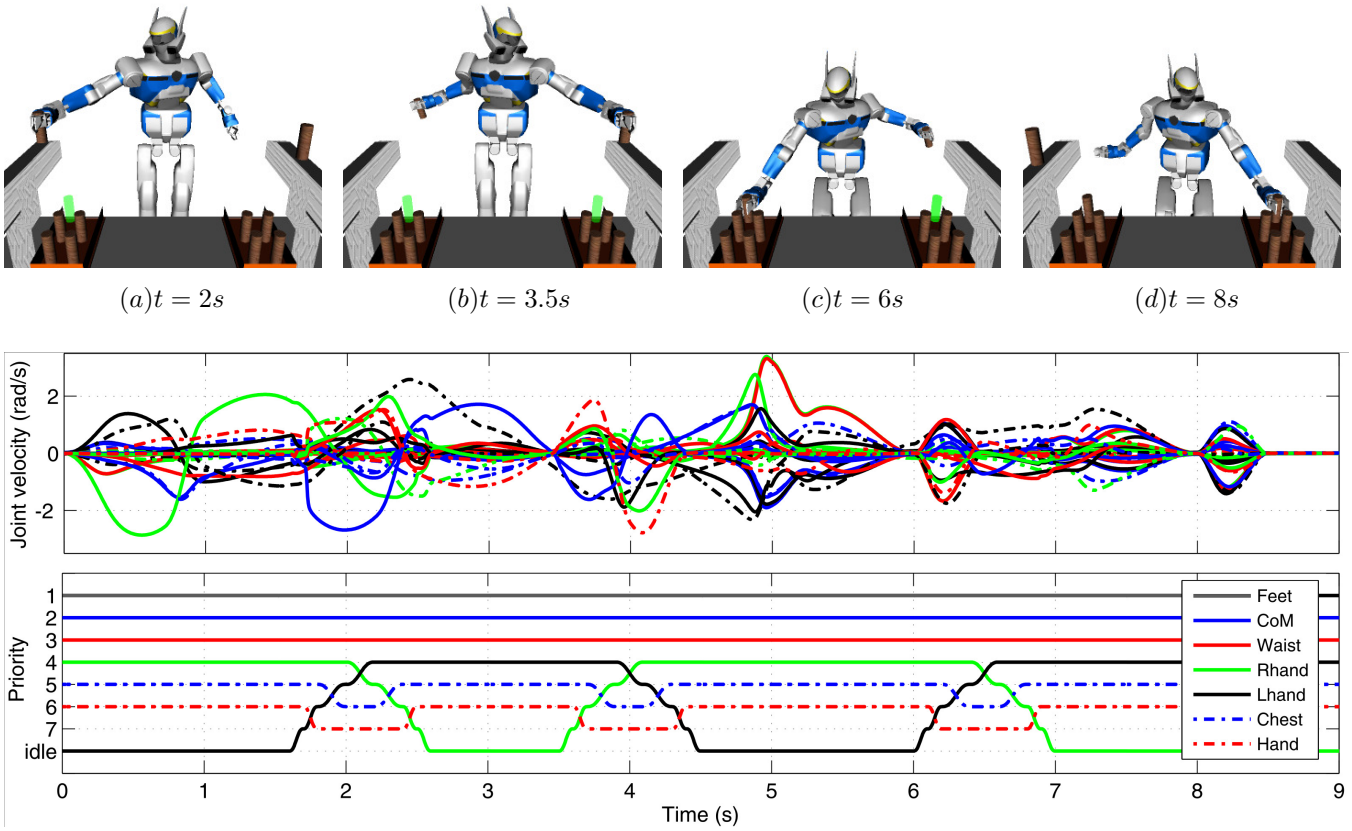

Fig. 3. **Top row**. A sequence of snapshots showing the postures of the robot when it reaches each target. **Second row**. The joint velocities profiles along the simulation. Note that they are smooth all the time. **Last row**. Order of the tasks according with their corresponding priority levels in the SoT during the simulation period.

[8] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 60–72, February 2007.

[9] F. Keith, P.-B. Wieber, N. Mansard, and A. Kheddar, "Analysis of the discontinuities in prioritized task-space control under discreet task scheduling operations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, USA, September 2011, pp. 3887–3892.

[10] J. Lee, N. Mansard, and J. Park, "Intermediate desired value approach for task transition of robots in kinematic control," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1260–1277, December 2012.

[11] P. Soueres, V. Cadenat, and M. Djeddou, "Dynamical sequence of multi-sensor based tasks for mobile robots navigation," in *7th IFAC Symposium on Robot Control*, Wroclaw, Poland, September 2003, pp. 423–428.

[12] F. Keith, N. Mansard, S. Miossec, and A. Kheddar, "Optimization of tasks warping and scheduling for smooth sequencing of robotic actions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, October 2009, pp. 1609–1614.

[13] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *IEEE International Conference on Advanced Robot*, Pisa, Italy, June 1991, pp. 1211–1216.

[14] P. Baerlocher and R. Boulic, "An inverse kinematic architecture enforcing an arbitrary number of strict priority levels," *The Visual Computer: International Journal of Computer Graphics*, vol. 20, no. 6, pp. 402–417, August 2004.

[15] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, USA, May 2010, pp. 3733–3738.

[16] C. V. Loan, "On the method of weighting for equality-constrained least-squares problems," *SIAM Journal of Numeric Analysis*, vol. 22, no. 6, pp. 851–864, October 1985.