OpenSoT: a Whole-Body Control Library for the Compliant Humanoid Robot COMAN

Alessio Rocchi, Enrico Mingo Hoffman, Darwin Caldwell, Nikos Tsagarakis

OpenSoT: a Whole-Body Control Library for the Compliant Humanoid Robot COMAN

Alessio Rocchi*, Enrico Mingo Hoffman*, Darwin G. Caldwell and Nikos G. Tsagarakis
Department of Advanced Robotics
Istituto Italiano di Tecnologia
Genova, Italy
*These authors contributed equally to this work

***Abstract*— A fundamental aspect of controlling humanoid robots lies in the capability to exploit the whole body to perform tasks. This work introduces a novel whole body control library called *OpenSoT*. *OpenSoT* is combined with joint impedance control to create a framework that can effectively generate complex whole body motion behaviors for humanoids according to the needs of the interaction level of the tasks. *OpenSoT* gives an easy way to implement tasks, constraints, bounds and solvers by providing common interfaces. We present the mathematical foundation of the library and validate it on the compliant humanoid robot COMAN to execute multiple motion tasks under a number of constraints. The framework is able to solve hierarchies of tasks of arbitrary complexity in a robust and reliable way.**

I. Introduction

In this paper we present a new whole body motion and compliance control architecture for the humanoid bipedal robot COMAN [1], Fig. 1, based on a task space whole-body trajectory generator called *OpenSoT*. *OpenSoT* is a novel control library that implements the idea of decoupling atomic tasks/constraints descriptions and solvers to execute multiple tasks and achieve complex motion behaviors. It employs a solver implementing a cascade of Quadratic Programming (QP) problems, and a set of tasks and constraints in velocity space in order to solve a generic hierarchical inverse kinematics problem on a humanoid robot. The generated velocities are integrated and sent to joint impedance controllers together with computed gravity compensation torques.

OpenSoT has been developed to address the motion and interaction control problems that the recent DARPA Robotics Challenge highlighted by integrating advanced and complex robotic systems, such as humanoids and other legged mobile manipulation machines, into real world scenarios where robots need to function and interact in unstructured and unknown environments. To cope with these scenarios, the control scheme should allow the robots to generate reliably complex and efficient motions at whole body level while accommodating/controlling the contact forces and interaction with the environment.

Typical methods and techniques to map task-space commands to joint commands can be classified as inverse kinematics or inverse dynamics schemes and can be implemented using a variety of low level controllers. Inverse dynamics schemes implemented on pure low level torque control [2] rely on the quality of the dynamic model [3] as well as on the performance of the joint torque sensing and regulation.

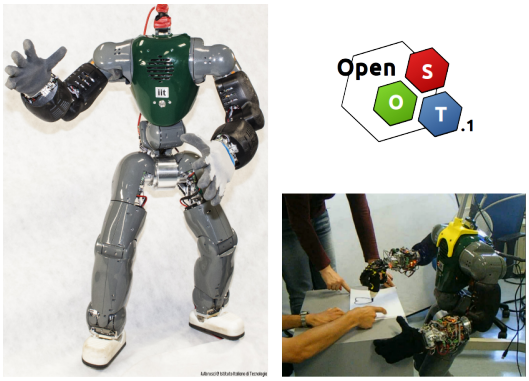

Fig. 1: COMAN is a 29 DOFs humanoid robot equipped with series elastic actuators

In the majority of the cases it is difficult to achieve adequate performance due to errors in all components including the model, the torque sensing accuracy and finally, torque tracking. On the other hand, classical stiff PID joint control is also unsuitable for tasks controlling interaction forces.

A trade-off between the two approaches can be represented by a combination of a centralized kinematic control working on top of a decentralized joint impedance control. The latter offers the robustness of decentralized control and adds compliance to the system allowing the possibility to regulate the impedance both at the joint level and at the Cartesian level through conservative congruence transformation [2]. Furthermore, it allows to introduce feed-forward torque terms based on inverse dynamics computations. We believe decentralized control schemes of this type are fundamental in reliable robots that must be able to recover from failures. An example is the case of a humanoid which tries to maintain balance when a recoverable hardware or software failure occurs: a decentralized architecture may then allow to keep a stable configuration while performing failure recovery procedures. *OpenSoT* implements these ideas. The goal was therefore to develop a high performance and flexible library that can generate reliably complex and efficient motions at whole body level. This yields the following features that we believe make the implementation of *OpenSoT* unique and attractive:

- Demonstrates adequate modularity through the separation of task descriptions, control schemes and solvers maximizing customization, flexibility and expandability.

- Provides user friendly interfaces for defining tasks, constraints and solvers to promote integration and cooperation in the emerging field of whole-body hierarchical control schemes.
- Demonstrates computation efficiency to allow for real time performance implementations.
- Allows ease of use and application with arbitrary robots through the Universal and Semantic Robotic Description Formats (URDF and SRDF).

The rest of the paper is structured as follows. Section II introduces the related literature on Whole-Body Inverse Kinematics/Dynamics compliant control, focusing mainly on schemes actually used in real hardware. Section III presents the global control scheme for compliant constrained IK control while Section IV describes the *OpenSoT* library. Finally we present experimental results obtained on COMAN in Section V. Conclusions are addressed in Section VI.

II. Related Works

Kinematic and dynamic inversion are well known problems in robotics. In general, given some tasks specified in Cartesian position, velocity, accelerations or forces, we want to find the joints positions, velocities, accelerations or torques that realize those tasks. Many solutions to this problem have been presented for single and multiple kinematic chains [4] and [5]. An interesting subset of these algorithms are those based on numerical optimization, since they allow for the explicit introduction of unilateral/bilateral constraints in the inverse kinematic/dynamic problem, which are fundamental when working on the real robot hardware [6].

In this brief review of the state of the art on full-body control, we will focus on some of the recent frameworks that have been used successfully on humanoid robots. One of the most famous framework oriented to whole-body control is the Stack of Tasks (SoT) from LAAS [7]. This work can be categorized under the group of the IK solvers with different implementations allowing to use both the classical pseudoinverse approach with null-space projection or decomposition methods that allow for one-shot solving of the whole stack (HCOD), to enforce lexycographic order in the stack. SoT has been used mostly in HRP-2 platform and our work is mainly inspired by this one. Another framework for whole-body task specification and control is iTaSC [8], with the current implementation supporting velocity-based whole-body control and equality constraints. It has been successfully demonstrated on the PR2 robot. In [9] IK is solved using a nonlinear program. Two kind of nonlinear optimizations are set up: a single-shot IK and trajectory optimization. This work was used in Atlas during the DRC trials coupled with a position control. A nonlinear optimization is also performed in [10] to perform IK on the iCub platform. There are many other frameworks including those that are based on Inverse Dynamics, implemented on top of a pure low level torque control (a notable example is [11]), yet it is difficult to find hardware platforms mature enough to implement control schemes of such frameworks and up to now no complex tasks have been experimentally demonstrated yet in humanoid bipedal robots.

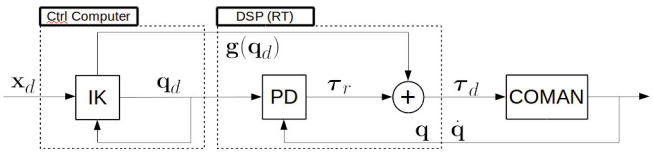

Fig. 2: Proposed control scheme

III. Control Scheme

In this section we introduce the Compliant Inverse Kinematics scheme as depicted in Fig. 2. The first block consists in a *Constrained Inverse Kinematic* engine, eventually in Closed Loop (CCLIK), that receives Cartesian references as input to perform a certain task defined in Task Space. This block generates joint position references and torques for gravity compensation for the second block that consists in a *joint impedance control*. In particular the *joint impedance control* is implemented in a decentralized way at each DSPs in the robot.

Decentralized Joint Impedance Control

The desired torque sent to actuator i is locally computed as

$$\tau_{d,i} = k_{q,i}(q_{d,i} - q_i) - k_{d,i}\dot{q}_i + g(q_d)_i \tag{1}$$

where q_i and $q_{d,i}$ are respectively actual and desired joint positions, $\dot{q}_i$ is the actual joint velocity, $k_{q,i}$ is a positive joint stiffness, $k_{d,i}$ is a positive joint damping and $g(q_d)_i$ is a gravity compensation torque computed at the desired joints configuration. The terms $q_{d,i}$ and $g(q_d)_i$ are computed by the CCLIK block. The control in (1) is implemented at the joint DSP level and is locally stable.

Inverse Kinematics

We consider a robot that executes n tasks simultaneously, and for each of these tasks T_i, a proper error function $e_i(q,t)$ is provided, describing the task error. The time derivative of the error can be computed as

$$\dot{e}_i = \frac{\partial e_i}{\partial q}\dot{q} + \frac{\partial e_i}{\partial t} = J_i\dot{q} + \frac{\partial e_i}{\partial t} \tag{2}$$

with J_i the error Jacobian. During the execution of a generic task it is desirable that the task error converges to 0, by imposing an exponential dynamic, that is

$$\dot{e}_i = -\lambda e_i \Rightarrow J_i\dot{q} = -\frac{\partial e_i}{\partial t} - \lambda e_i = \dot{e}_i^* \tag{3}$$

If the robot is redundant with respect to a task, secondary tasks can be also added and executed without affecting the performance of the primary task, and given a set of tasks described by the couple $T_i = (J_i, \dot{e}_i^*)$, the robot can be commanded to execute them using its whole body motion capabilities. In order to implement the method and obtain this result, the relative importance between tasks needs first to be defined. Thus, two kinds of relationships: *hard priority* and *soft priority* needs to be set. A task has *hard priority* with respect to another task if the latter can not deteriorate the

solution of the first one. *Soft priorities* are defined between tasks at the same level so all the solutions are influenced by each other proportionally to theirs weights.

The execution of a set of tasks has a well-known solution in the stack of tasks, where hard priorities are enforced by the order of the task in the stack. To take into account also soft priorities the augmented Jacobian formulation [12] can be employed. It must be noted though that the augmented Jacobian formulation alone cannot enforce hard priorities since adding many tasks together can generate an ill-conditioned augmented Jacobian matrix. Therefore, to generate whole-body motions, a series of QP problems in cascade is instead solved [13]. This is a well known method to derive motions by executing tasks adding bilateral constrains to the inverse kinematics problem [6]. A generic task can be described as

$$\begin{aligned} \dot{q}_1 = \quad & \underset{\dot{q}}{\text{argmin}} \quad \|J_i\dot{q} - \dot{e}_i^*\| \\ & s.t. \quad A_{c,1}\dot{q} \leq \mathrm{b}_{c,1} \end{aligned} \tag{4}$$

The formulation used in (4) for the constraints can be profitably used to express lower and upper bounds for the variable value as well as equality constraints.

In general, the n^{th} task will then be written as

$$\begin{aligned} \dot{q}_d = \quad & \underset{\dot{q}}{\text{argmin}} \quad \left\|J_n\dot{q} + \lambda e_n + \tfrac{\partial e_n}{\partial t}\right\| \\ & s.t. \qquad A_1\dot{q} = A_1\dot{q}_1 \\ & \qquad\qquad \vdots \\ & \qquad A_{n-1}\dot{q} = A_{n-1}\dot{q}_{n-1} \\ & \qquad A_{c,1}\dot{q} \leq \mathrm{b}_{c,1} \\ & \qquad\qquad \vdots \\ & \qquad A_{c,n}\dot{q} \leq \mathrm{b}_{c,n} \end{aligned} \tag{5}$$

where $\dot{q}_d$ is the desired velocity (control variable). In (5) the previous solutions $\dot{q}_i, i < n$ are taken into account with constraints of the type $A_i\dot{q} = A_i\dot{q}_i \ \forall i < n$, so that the optimality of all higher priority tasks is not changed by the current solution. While in (5) the first task has a relationship of hard priority with respect to the second, and so on, for each *level* of priority, a soft priority relationship between tasks can be imposed introducing the relative weights β_i, so that the augmented Jacobians and the error vectors can be written as

$$\begin{aligned} J_{\text{aug}} &= \begin{bmatrix}\beta_1 J_1^T & \dots & \beta_n J_n^T\end{bmatrix}^T \\ e_{\text{aug}} &= \begin{bmatrix}\beta_1 e_1^T & \dots & \beta_n e_n^T\end{bmatrix}^T \end{aligned} \tag{6}$$

where the soft priority between tasks is altered by tuning the relative weights β_i, with higher priority tasks having larger β_i. As already mentioned, in this case the tasks can still influence each other performance. A mixture of hard and soft priorities is in general needed to describe a stack of tasks. The solution obtained can then be sent directly to a velocity controlled robot or integrated in a position controlled robot as

$$q_d = q + \dot{q}\delta t \tag{7}$$

where δt is the control loop period.

IV. OpenSoT

OpenSoT is a robotics library tool oriented to Whole-Body Control. The main idea behind its implementation is to decouple the task description, and the solver used to solve the general Inverse Kinematics/Dynamic problem. This distinction allows to easily switch the control, even with the same task description, by using a different solver. *OpenSoT* aims at providing a standard way to describe the aforementioned entities in an atomic way, and at the same time to build a repository of common entities that can be reused to create a generic stack using a user friendly development and integration interface.

OpenSoT offers classes to describe different tasks and constraints and to implement different solvers oriented to a particular control type (velocity, acceleration, torque). According to the control type, different controllers are available in literature. Defining a task once in a proper way still allows to switch between different control laws, as shown in [14], thus decoupling task description and type of control. The same applies to constraints and bounds [15].

OpenSoT Tasks

In *OpenSoT*, a task T_i is in general defined as:

$$T_i = (J_i, \quad \dot{e}_i^*) \tag{8}$$

including a set of constraints for that task. Tasks are implemented through the class *Task* which provides an interface to obtain A and b, a weight matrix W and a scalar weight λ for the task (`getA()`, `getb()`, `getWeight()`, `getlambda()`). In our framework it is possible to *augment* a task with an operation called *Aggregation*.

Aggregated: The aggregated task constructs an augmented Jacobian starting from a definition of more basic tasks as

$$T_{\text{agg}} = \quad \left(\begin{bmatrix}A_1^T & A_2^T\end{bmatrix}^T, \quad \begin{bmatrix}b_1^T & b_2^T\end{bmatrix}^T\right) \tag{9}$$

Cartesian: It is possible to define a general Cartesian task as the *aggregate* of a Cartesian position task and a Cartesian orientation task. The Cartesian task computes the (relative) Jacobian between any given base and distal links in the kinematic tree, ${}^b\mathrm{J}_d$. The Cartesian errors in position and orientation are computed respectively as:

$$\begin{aligned} e_p &= p_d - p \\ e_o &= -(\eta_d\epsilon - \eta\epsilon_d + [\epsilon_d\times]\epsilon) \end{aligned} \tag{10}$$

and the task is defined as:

$$\begin{aligned} T_{C,p} &= \left({}^b\mathrm{J}_{d,p}, \quad \dot{p}_d + K_p e_p\right) \\ T_{C,o} &= \left({}^b\mathrm{J}_{d,o}, \quad \omega_d + K_o e_o\right) \end{aligned} \tag{11}$$

where $p_d = [x_d \quad y_d \quad z_d]$ is the desired position and $\alpha_d = [\eta_d \quad \epsilon_{1,d} \quad \epsilon_{2,d} \quad \epsilon_{3,d}]$ is the desired orientation expressed as a quaternion [14], K_p and K_o are positive definite matrices and $\xi_d = [\dot{p}_d \quad w_d]$ is the desired Cartesian velocity for the end-effector. A particular case is the *CoM* task which is defined as a Cartesian position task.

Postural: A postural task is defined in joint space as:

$$T_p = (\mathrm{I}, \quad \dot{q}_d + K_q(q_d - q)) \tag{12}$$

Minimum Effort: The minimum effort task is again defined in joint space as:

$$T_g = \left(\mathrm{I}, \quad \alpha_g \nabla(g(q)^T g(q))\right) \tag{13}$$

where the gradient is computed numerically by means of two-point estimation and the Hessian is set to identity as in the *Gradient Projection Method* [16].

Constraints and Bounds

Constraints model equalities and bilateral/unilateral inequalities using the simple form:

$$C = (A_c, \quad b_c) \tag{14}$$

Bounds are a particular kind of constraints, applied to all the tasks in the stack, where A_c is always the identity. We can specify lower and upper bounds as

$$b = (l_b, \quad u_b) \tag{15}$$

Constraints and bounds are implemented in *OpenSoT* through the class *Constraint* which implements a simple interface providing equality constraints (`getAeq()`, `getbeq()`), inequality constraints (`getAeineq()`, `getbLowerBound()`, `getbUpperBound()`) and bounds (`getLowerBound()`, `getUpperBound()`).

Aggregated: The aggregated constraint performs merging of a list of constraints into a single constraint, in particular piling the equalities and inequalities constraints, and merging the bounds as follows:

$$C_{\text{agg}} = \left(\begin{bmatrix} A_{c,1}^T & A_{c,2}^T \end{bmatrix}^T, \quad \begin{bmatrix} b_{c,1}^T & b_{c,2}^T \end{bmatrix}^T\right) \tag{16}$$

$$b_{\text{agg}} = \left(\max(\mathrm{l}_1, \mathrm{l}_2), \quad \min(\mathrm{u}_1, \mathrm{u}_2)\right) \tag{17}$$

ConvexHull: CoM is bounded to lay inside the convex hull defined by the contacts with the environment, where we can write

$$C_{\text{CH}} = \left(\begin{bmatrix} a_0 & b_0 \\ \vdots & \vdots \\ a_{n-1} & b_{n-1} \end{bmatrix}, \quad \begin{bmatrix} -c_0 \\ \vdots \\ -c_{n-1} \end{bmatrix}\right) \tag{18}$$

with a_i,b_i,c_i coefficients of the implicit equation of the line $a_i x + b_i y + c_i = 0$, bounding the convex hull. These lines are expressed in a frame attached to the CoM and parallel to the inertial frame, and are obtained by finding the coefficients of a line passing through two consecutive points of the convex hull of the support polygon. The convex hull is obtained by creating a hull of a point cloud of contact points between the foot and the ground, which can be obtained by skin sensors or, in current implementation, by the foot model assuming full-foot contact with the ground.

CartesianLinearVelocity: This task implements limits on the Cartesian velocity of any link w.r.t. another link, or the inertial frame

$$C_{\text{C,p}} = \left({}^b\mathrm{J}_d, \quad {}^b\mathrm{v}_{\text{max,d}}\right) \tag{19}$$

The constraints *CoMVelocity* are a peculiar case where instead of the distal link we use the *CoM*.

TABLE I: Solver Time

# Stacks	Tasks	Time [s]
2	postural, CoM	0.003
3	postural, CoM, left_arm	0.005
4	postural, CoM, left_arm, right_arm	0.007

JointLimits:

$$b_{\text{JointLimits}} = (\mu(q_{\min} - q), \quad \mu(q_{\max} - q)) \tag{20}$$

JointVelocity:

$$b_{\text{JointVelocity}} = (-\alpha_i \dot{q}_{\max} \Delta t, \quad \alpha_i \dot{q}_{\max} \Delta t) \tag{21}$$

with $\mathrm{u}_{\text{j\_lims}}$ and $\mathrm{l}_{\text{j\_lims}}$ the $\dot{q}$ where $\alpha_i \leq \alpha_{i+1}$ scales the bounds in order to implement a simple velocity allocation scheme between tasks at different priority levels.

OpenSoT Solver

OpenSoT provides a class to implement different *Solvers* that uses the classes *Tasks*, *Constraints* and *Bounds* to solve the IK problem for a desired control type. The actual solver supported in *OpenSoT* is based on (5) and is currently supporting only velocity based task and Constraints. Therefore *velocity control* is the available control type in the current state of implementation while other type of solvers which support different control modalities are due to be integrated soon. The solver uses qpOASES [17], an open-source C++ implementation of an on-line active set strategy [18], part of the ACADO suite. qpOASES implements an automatic regularization technique (therefore there is no need for explicit singularity avoidance and escaping) and warm-start functionalities. The solver permits to define any type of stack and it handles the bounds in a global way. To test the performances of the solver we have prepared a benchmark were a variable sized stack is created and solved. The lowest priority task is always a postural task (12) while the others are Cartesian tasks (11), bounds on joint limits and velocities are also added to the optimization. Table I shows the results to solve one control step of optimization[1] for 29 variables.

V. Experiments

To evaluate the performance of *OpenSoT*, experiments were performed using a compliant humanoid robot. Apart from demonstrating the implementation and fundamental functionality of the library, the experiments were targeting at demonstrating also the effect of changing on-line the low-priority joint space task from pure *postural* to pure *minimum effort*.

Control Scheme

In the experiment a *Control Computer* runs the whole-body inverse kinematic framework in open-loop. Gravity compensation terms are derived at each desired configuration computed by *OpenSoT* and are used by the joint-level impedance controller running inside the motor driver DSPs. The Joint active stiffness and damping are kept constant during the experiment. In the experiment the joint impedance control is enabled only in the upper body/arms, while a joint position control is used in the lower body.

[1]These results were obtained using a Intel Core i7 with 6GB of RAM

Tasks Description

The stack is composed by two tasks, where J_1 is an *Aggregated Task* composed of *Cartesian* tasks that control the hands and swing foot end-effectors, and a *CoM* task to control the Center of Mass considering the whole body. Using the augmented Jacobian[19] formulation provided by `Aggregated`, we have

$$\begin{aligned} J_1 &= \begin{bmatrix} J_{\text{l\_wr}}^T & J_{\text{r\_wr}}^T & J_{\text{CoM}}^T & J_{\text{sw\_ft}}^T \end{bmatrix}^T \\ e_1 &= \begin{bmatrix} e_{\text{l\_wr}}^T & e_{\text{r\_wr}}^T & e_{\text{CoM}}^T & e_{\text{sw\_ft}}^T \end{bmatrix}^T \end{aligned} \tag{22}$$

where $J_{\text{l\_wr}}$ and $J_{\text{r\_wr}}$ are the (6×29) Jacobians of the left and right arms respectively computed w.r.t. the *base link* of the robot. Notice that $J_{\text{l\_wr}}$ and $J_{\text{r\_wr}}$ share the torso kinematic chain. J_{CoM} is the (3×29) Jacobian of the CoM of the robot computed w.r.t. the *left foot* (support foot) and finally $J_{\text{sw\_ft}}$ is the (6×29) Jacobian of the *right foot* (swing foot) computed w.r.t. the *left foot*. The second task is again an *Aggregated* in joint space needed to stabilize auto-motions due to redundancy, which aggregates a *MinimumEffort* and *Postural* task. Again, the solution of the second task lies in the null-space of the first QP problem, meaning the residuals of the first task remain constant after solving the second

$$\begin{aligned} J_2 &= \begin{bmatrix} \beta I_{\text{nJ}}^T & (1-\beta)\, I_{\text{nJ}}^T \end{bmatrix}^T \\ e_2 &= \begin{bmatrix} \beta \lambda_{11} b_p^T & +(1-\beta)\, \lambda_{12} b_g^T \end{bmatrix}^T \end{aligned} \tag{23}$$

The parameter $\beta \in (0, 1)$ allows to smoothly weight between a *postural* joint space task to *minimum effort* joint space task. Each level of the stack is bounded by an *Aggregated* constraint containing one *JointLimits Bound* and a *JointVelocity Bound*. The CoM is not controlled in a higher level task, meaning we do not wish to control its exact position, rather we set an admissible region bounded by the convex hull by imposing fixed margins from the convex hull vertices. This is done by creating a constraint of type *ConvexHull*, and a constraint of type *CoMVelocity* to constraint the maximum Cartesian speed of the CoM. The optimization algorithm uses the `reliable` default settings of qpOASES, with regularisation enabled and regularisation ϵ set as `epsRegularisation*=2E2`, the maximum number of working set calculations `nWSR=32`. A reference configuration for the postural task q_{ref} and a reference pose for all the controlled end-effectors is set by sensing their respective values during the start-up phase of the control. During the experiment, the gain β is tuned manually on-line. Speed of execution is obtained by using the warm-start method, which allows to have an initial guess for the active-set in the QP solver. This allows to solve this stack within 4ms.

Results

Fig. 3 presents the 2-norm of the torques in all the 29 joints of the robot. It is easy to recognize the different part where the last task is pure postural ($\beta = 1.0$) and where the last task is pure min effort ($\beta = 0.0$). The difference between the 2-norm of the torques in pure postural task and in pure min effort task is around $2[Nm]$ for the given references. The second figure in Fig. 4 depicts the 2-norm of CoM position error. The error for a certain kinematic chain is computed as the sum of the 2-norms of 10 between the desired and computed Cartesian pose. It can been seen that the CoM is adequately maintained with the predefined reference position even during the switching of the tasks. Error in joint trajectory generation increases only during switching phases. The 2-norm of the vector of the joint errors is presented in the first figure in Fig. 4. Finally in Fig. 5 the COMAN is shown in different postures arising by changing β in the secondary task. This particular stack of tasks, constraints and bounds have been also used to perform tasks with high level of interaction with the environment such as a writing task, Fig. 6 and a cleaning task, Fig. 7. The joint impedance control makes the interaction possible and safe reducing impact/contact forces that may compromise the robot stability, damage the robot itself or the environment. The whole-body IK runs in open loop to prevent the action of the feedback during interaction from counteracting the effect of compliant behavior.

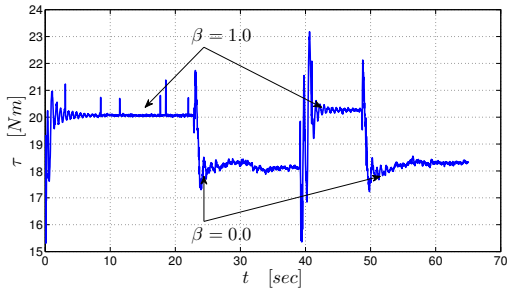

Fig. 3: 2-norm of torques during switching from a pure postural task to a mix of postural and minimum effort task

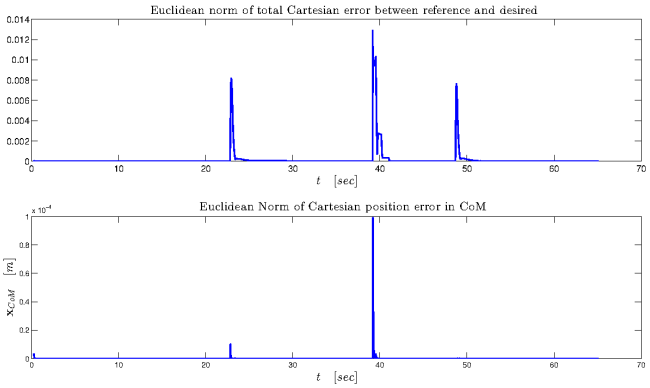

Fig. 4: 2-norm of Cartesian errors for CoM and end effectors

VI. Conclusions

This work introduced a novel whole-body control library called *OpenSoT*s. *OpenSoT* is combined with joint impedance control to create a framework tool that can effectively generate complex whole-body motion behaviors for humanoid robots. Such framework allows to control the robot by solving a hierarchy of tasks with linear constraints. The

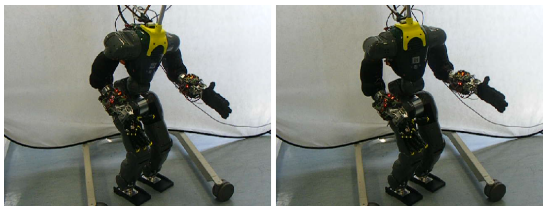

Fig. 5: COMAN holding end-effectors pose while executing a postural task (left) or a minimum effort task (right)

Fig. 6: COMAN performing a drawing task on a desk, interaction is handled by joint impedance control

framework provides an easy way to implement new tasks, constraints, bounds and solvers by providing common interfaces. The theoretical foundation of *OpenSoT* was presented and its performance was demonstrated in a multi-objective task in which the motion of all end-effectors of the robot and the centre of mass (CoM) were controlled while taking into account joint limits and joint velocity limits. At the same time, a minimum-effort task in the null-space of these tasks is imposed at lower priority. Compliant behavior was obtained through a low level joint impedance control with gravity compensation at the desired configuration. *OpenSoT* is released free and open source at `https://github.com/robotology-playground/OpenSoT`.

VII. Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7-ICT-2013-10] under grant agreements n.611832 WALKMAN and ERC Advanced Grant no. 291166 SoftHands.

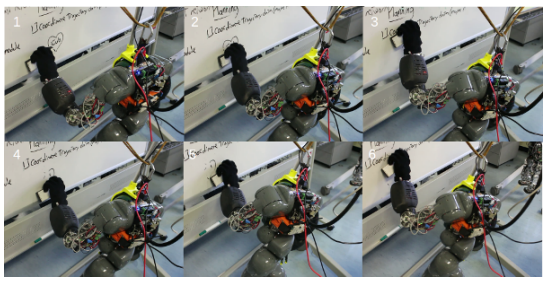

Fig. 7: COMAN erasing a whiteboard, interaction is handled by the joint impedance control

References

[1] A. Ajoudani, J. Lee, A. Rocchi, M. Ferrati, E. M. Hoffman, A. Settimi, D. G. Caldwell, A. Bicchi, and N. G. Tsagarakis, "Manipulation framework for compliant humanoid coman: Application to a valve turning task," in *IEEE-RAS International Conference on Humanoid Robots, Humaoids 2014*, (Spain), In Press.

[2] C. Ott, *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*. Springer Publishing Company, Incorporated, 1 ed., 2008.

[3] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Rob. Autom.*, vol. 3, no. 1, pp. 43–53, 1987.

[4] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, no. 3, pp. 201–212, 1990.

[5] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 3103–3109, IEEE, 2013.

[6] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, pp. 1006–1028, Jun 2014.

[7] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *International Conference on Advanced Robotics (ICAR)*, p. 119, June 2009.

[8] R. Smits, H. Bruyninckx, and J. De Schutter, "Software support for high-level specification, execution and estimation of event-driven, constraint-based multi-sensor robot tasks," in *Proceedings of the 2009 International Conference on Advanced Robotics*, (Munich, Germany), 2009.

[9] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. Perez D'Arpino, R. Deits, M. DiCicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K.-T. Yu, J. Shah, K. Iagnemma, R. Tedrake, and S. Teller, "An architecture for online affordance-based perception and whole-body planning," *Journal of Field Robotics*, 2014.

[10] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots.," in *IROS*, pp. 1668–1674, IEEE, 2010.

[11] 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, 2010.

[12] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *The International Journal of Robotics Research*, vol. 10, no. 4, pp. 410–425, 1991.

[13] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *Robotics, IEEE Transactions on*, vol. 27, no. 4, pp. 785–792, 2011.

[14] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," no. 6, pp. 737–757, 2008.

[15] F. Flacco, A. D. Luca, and O. Khatib, "Motion control of redundant robots under joint constraints: Saturation in the null space," in *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pp. 285–292, 2012.

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

[17] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, pp. 1–37, 2013.

[18] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[19] A. De Santis, G. Di Gironimo, L. Pelliccia, B. Siciliano, and A. Tarallo, "Multiple-point kinematic control of a humanoid robot," in *Advances in Robot Kinematics: Motion in Man and Machine*, pp. 157–168, Springer, 2010.