FP7-600716

Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

D3.3
Local solver in compliant and unforeseen cases

Editor(s)	Vincent Padois[1]
Responsible Partner	UPMC
Affiliations	[1] Sorbonne Universités, UPMC Paris 06, UMR CNRS 7222, Institut des Systèmes Intelligents et de Robotique (ISIR), F-75005, Paris, France.
Status-Version:	Final-1.0
Date:	Feb. 28, 2017
EC Distribution:	Consortium
Project Number:	600716
Project Title:	Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

Title of Deliverable:	Local solver in compliant and unforeseen cases
Date of delivery to the EC:	28/2/2017

Workpackage responsible for the Deliverable	WP3
Editor(s):	Vincent Padois
Contributor(s):	Ryan Lober, Francesco Nori, Vincent Padois, Daniele Pucci, Francesco Romano, Olivier Sigaud, Silvio Traversaro
Reviewer(s):	
Approved by:	All Partners

Abstract	Highly redundant robots, such as humanoids, can execute multiple simultaneous tasks allowing them to perform complex whole-body behaviors. Unfortunately, tasks are generally planned without close consideration for the underlying controller being used, or the other tasks being executed. Because of this, tasks are often incompatible with one another and/or the system constraints, and cannot always be accomplished simultaneously. These incompatibilities can be managed using prioritization and gains, but tuning them is tedious. After recalling the structure of the whole-body controller that will be used for the final year demonstration (originally presented in Deliverables 3.2 [1] and 5.3 [2]), this deliverable introduces and develops the concept of task compatibility optimization, as a alternative/complementary method to the prioritization learning methods introduced in Deliverable 4.3 [3]. Task compatibility optimization automatically improves task compatibility by modifying their trajectories using reinforcement learning. To do so, the tasks are iteratively optimized by minimizing a compatibility cost, which measures the compatibility between one or more tasks, and the system constraints. Using two common, CoDyCo related, scenarios, it is shown that that task compatibility optimization results in whole-body behaviors which better match the original intent of the task combination without the need for manual tuning of task/controller parameters, heuristics, or re-planning.

Keyword List:	Whole-body controllers, Multi-contacts, Unforseen situations, Task compatibility optimization, Learning and adaptation

Document Revision History

Version	Date	Description	Author
v. 1.0	Feb. 28, 2017	Final version	Vincent Padois

Table of Contents

Index of Figures

Foreword: Research works performed within the framework of CoDyCo and that directly relate to this deliverable are cited as **[xx]**.

1 Introduction

Highly redundant robots, such as humanoids, can execute multiple simultaneous tasks allowing them to perform complex whole-body behaviors. Unfortunately, tasks are generally planned without close consideration for the underlying controller being used, or the other tasks being executed. Because of this, tasks are often incompatible with one another and/or the system constraints, and cannot always be accomplished simultaneously. These incompatibilities can be managed using prioritization and gains, but tuning them is tedious.

After recalling the structure of the whole-body controller that will be used for the final year demonstration (originally presented in Deliverables 3.2 [1] and 5.3 [2]), this deliverable introduces and develops the concept of task compatibility optimization, as a alternative/complementary method to the prioritization learning methods introduced in Deliverable 4.3 [3]. Task compatibility optimization automatically improves task compatibility by modifying their trajectories using reinforcement learning. To do so, the tasks are iteratively optimized by minimizing a compatibility cost, which measures the compatibility between one or more tasks, and the system constraints. Using two common, CoDyCo related, scenarios, it is shown that that task compatibility optimization results in whole-body behaviors which better match the original intent of the task combination without the need for manual tuning of task/controller parameters, heuristics, or re-planning

2 Controller structure

The remainder of this deliverable is mostly based on the recently submitted paper **[4]**.

3 Task compatibility optimization: motivations

Modern control architectures employ multiple levels of control in order to decouple complex behaviors into manageable control problems. As recalled in section 2, at the lowest level is reactive whole-body control, where joints torques are calculated at high frequency (~ 1kHz) given one or more tasks [5]. As presented in Deliverable 3.2 [1], the control problem can be written as a constrained convex optimization, where the objective function is a combination of task errors, and the constraints are the equations of motion, articulation and actuation limits, and contacts [6, 7, 8]. Task errors are calculated as the difference between the current task state and its reference value. This reference value comes from the next level of task servoing. At this level, closed loop controllers are used to servo task trajectories using state feedback (PID) or Model Predictive Control (MPC) schemes at frequencies between 100Hz and 10Hz **[9]**, [10], **[11]**. These task trajectories are provided by higher-level open-loop planning which takes seconds to minutes, and generally combines operator expertise and automated planning algorithms [12, 13]. This control hierarchy of planning, servoing, and whole-body control is

presented in Fig. 1.

Because each level in the control hierarchy is agnostic of the others by design, there is no guarantee that the planned task trajectories will be executed properly by the lower control layers **[14, 15]**. Furthermore, even though specific contexts such as non rigid contacts may be accounted for as described in Deliverable 3.2 [1], tasks may conflict with one another or be infeasible with the system constraints [16, 17]. The end result is typically unstable or undesirable whole-body behaviors, and these tasks can be qualified as *incompatible*. Prioritization techniques presented in Deliverable 3.2 [1] use weighted sums[6, 8], hierarchies [7, 18, 19] or a mix of both [20, 21] to manage task incompatibilities at the whole-body control level, but are difficult to tune and only hide the problem. Moreover, tasks incompatibilities may be temporal and change over the course of the movement so applying static priorities may be overly restrictive. While the research works presented in Deliverable 4.3 [3] provide very powerful tools to reactively adapt priorities [22], learn proper prioritizations **[23, 24]** or derive them from demonstrations **[25]**, it seems obvious that well designed tasks should not need to be prioritized.

Given that it is the task reference values which generate the incompatible control optima, an alternative to prioritization tuning is to modify the task trajectories supplied by planning and make them compatible as initially suggested in **[26]**. To do so, a feedback loop must be implemented, which measures the errors induced by incompatibilities and changes the task trajectories to reduce them. It should also take into account the servoing and whole-body control levels with all of their parameters, as well as the robot's dynamics and environment. Given the complexity of the proposed feedback loop, one solution is to use model-free Reinforcement Learning (RL) techniques to modify the trajectories through trial and error by minimizing some cost function using Black-Box Optimization (BBO) solvers [27].

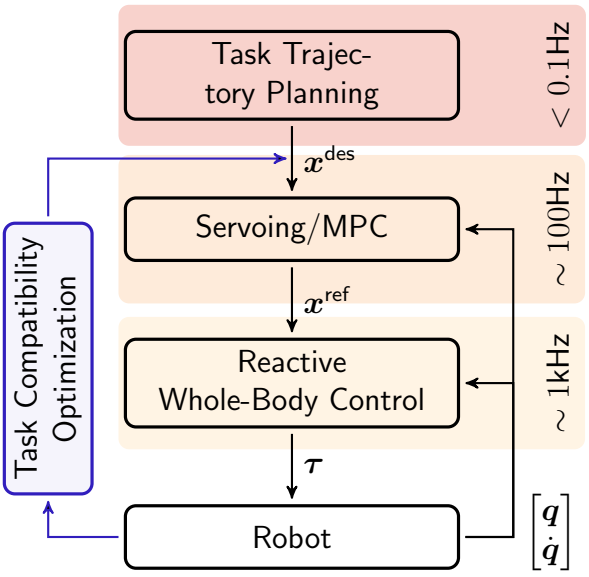


Figure 1: A modern control hierarchy for highly redundant robotic systems, e.g. humanoid robots. At the lowest level is whole-body control, which determines the torques needed to accomplish a set of tasks. At the intermediate level, these tasks are controlled by the servoing/MPC level where task trajectory errors are compensated using state feedback. Finally the task trajectories are provided by high-level planning, which is usually a combination of operator expertise and automated planning. Each of these levels operates independently from one another and a feedback mechanism is needed to measure and compensate for tasks which are not executed as planned. This is the role of the Task Compatibility Optimization loop proposed in this paper.

The objective of this study is to establish the task compatibility optimization loop, shown

on the left in Fig. 1, by iteratively improving task trajectories using RL. To do so, some details are provided on how task trajectories are parameterized providing variables with which they can be modified. A generic task compatibility cost is then developed from simple principles which measures the incompatibility between one or more tasks and the robot's constraints. Using two common BBO solvers, this compatibility cost is minimized by optimizing the task trajectory parameters. This task compatibility optimization is then tested on two typical muti-task scenarios. In the first scenario the relatively banal chore of reaching while balancing is studied. While seemingly simple, reaching is a key ingredient in robot autonomy, which often requires parameter and gain tuning before done reliably. A performance comparison of two BBO solvers for this experiment is presented to illustrate the generality of the framework. The second experiment explores the dynamically complex activity of moving from sitting to standing. This motion requires contact breaking and potentially unstable dynamic equilibrium to succeed. In both experiments a Center of Mass (CoM) task is used to maintain balance, and its trajectory is optimized to minimize the task compatibility cost. Through these two completely different motion scenarios, the proposed generic task compatibility optimization loop is shown to dramatically improve task achievement, without ever touching the low-level control parameters.

4 Methods

In this section, the methods and tools used to develop the task compatibility optimization are described. First, trajectory parameterization is detailed. A task compatibility cost is then developed to measure the degree of incompatibility between one or multiple tasks. A brief overview of the two BBO solvers used is provided. Finally the use of these components to optimize task compatibility is explained.

4.1 Task Parameterization

For the purpose of brevity, here, Cartesian acceleration tasks only are considered. In the whole-body controller used in this study, [6], an acceleration task error, T_i, is formulated as,

$$T_i = \left\| J_i(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}_i(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \ddot{\boldsymbol{x}}_i^{\text{ref}} \right\|^2 , \tag{1}$$

where J_i and $\dot{J}_i$, are the task Jacobian and its derivative, $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$, the joint-space state variable and $\ddot{\boldsymbol{x}}_i^{\text{ref}}$ the reference operational-space acceleration. The $\ddot{\boldsymbol{x}}_i^{\text{ref}}$ values are provided by task servoing, which are feedforward proportional-derivative controllers. These trajectories are generated from a series of keyframes/waypoints, which represent task coordinates of particular importance. A single position waypoint is given by $\boldsymbol{\lambda}_i = \boldsymbol{x}_i = \begin{bmatrix} x & y & z \end{bmatrix}_i^T$, while a set of n_λ waypoints is denoted $\Lambda = \begin{bmatrix} \boldsymbol{\lambda}_1 & \boldsymbol{\lambda}_2 & \dots & \boldsymbol{\lambda}_n \end{bmatrix}$. Given Λ, a variety of methods exist for generating a trajectory, e.g. splines, polynomials, optimal control methods, etc. In this study the time-optimal formulation proposed by [28] is used, which can be passed Λ and returns a time-optimal trajectory through the waypoints, with a duration, d_Λ, dependent on the velocity and acceleration limits imposed on the movement.

4.2 Task Compatibility Cost

In this section a measure of task compatibility is derived from simple principles related to the task error objective function in (1).

4.2.1 Tracking Cost

In (1), $\ddot{\boldsymbol{x}}_i^{\text{ref}}$ is the optimal operational-space value for T_i at time, t. If the objective is perfectly realized then the squared norm error is zero, meaning that the robot perfectly follows the task's operational-space reference. Therefore, any error in the position tracking reflects an imperfect optimization of the squared norm task error and consequently a task incompatibility. Using this concept, a *tracking cost* is defined as

$$j_t^i = \sum_{t=0.0}^{t_{\text{end}}} \|\boldsymbol{x}_i(t) - \boldsymbol{x}_i^{\text{ref}}(t)\|^2 \ , \tag{2}$$

where $\boldsymbol{x}_i(t)$, is the task frame position and $\boldsymbol{x}_i^{\text{ref}}(t)$, its reference, at time t. The term t_{end} is the actual total duration of the whole-body motion.

4.2.2 Goal Cost

The assumption is made that the ultimate objective of any point to point trajectory is to reach its target coordinate, or final waypoint. With this in mind a *goal cost* is developed as,

$$j_g^i = \sum_{t=0.0}^{t_{\text{end}}} \frac{t}{d_\Lambda} \|\boldsymbol{x}_i(t) - \boldsymbol{\lambda}_n\|^2 \ , \tag{3}$$

where $\boldsymbol{x}_i(t) - \boldsymbol{\lambda}_n$, is the difference between the task's position at t and the final waypoint in its trajectory. The weight of this difference increases linearly from 0.0 with time.

4.2.3 Energy Cost

Finally, given the redundancy of the system, it is possible that many whole-body motions exist for the execution of the same set of n_{tasks} tasks to be performed. To reduce these possibilities, those which are energy optimal are favoured using an *energy cost*,

$$j_e = \beta \sum_{t=0.0}^{t_{\text{end}}} \|\boldsymbol{\tau}(t)\|^2 \ , \tag{4}$$

where the term β is used to scale the energy cost for meaningful comparison with j_t and j_g. Here, $\beta = 1.0\text{e}{-4}$ is used.

4.2.4 Compatibility Cost

The compatibility cost for n_{tasks} tasks can be calculated by first summing their tracking and goal costs, then adding the energy cost, which is common to all tasks, and finally averaging

over t_{end},

$$j_c = \left[j_e + \sum_{i=1}^{n_{\text{tasks}}} \left(j_t^i + j_g^i \right) \right] \Big/ t_{\text{end}} \quad . \tag{5}$$

With (5) the compatibility cost of an arbitrary number of tasks can be estimated. This cost, however, has no absolute significance on its own. There is no threshold value for determining if a set of tasks is compatible, incompatible, or somewhere in between. For any task set, the j_c of the initial task trajectories, denoted j_c', are taken as the reference with which all other task trajectories are compared using,

$$j_c^{\text{new}} = \frac{j_c}{j_c'} \, . \tag{6}$$

This means that for any task combination, the initial task trajectories have a compatibility cost equal to 1.0. Any modifications to the trajectories yielding a $j_c^{\text{new}} < 1.0$ represent improvements in task compatibility, and vice versa for $j_c^{\text{new}} > 1.0$.

Finally tasks used for computation of (5) must be determined. In a typical whole-body controller there may be a multitude of simultaneously active tasks at any given time. Generally there are a few primary tasks which are designed to affect some motion, and any number of helper tasks used to improve the stability, posture, or quality of the movement. The decision of which tasks to use to calculate (5) is an open one, but here the principal tasks are used, such as those for balance and reaching.

4.3 BBO Solvers

The task compatibility optimization problem is non-linear, non-convex, and possibly discontinuous. BBO solvers are therefore used. Local solvers, such as Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) use the statistics from a set of objective variable samples and their costs to estimate the mean and covariance of the sample distribution and then update this distribution in the direction of the natural gradient [29]. Global solvers such as Bayesian Optimization (BO) explicitly model the latent cost function using Gaussian Processes and provide a set of parameters to test which both minimize the expected cost and the uncertainty of the cost model [30].
BO solvers usually require fewer trials to obtain an optimal solution and have become a popular choice in robotics because of this efficiency [31, 32, 33]. The performance and solution quality of BO and CMA-ES are highly dependent on proper tuning of the solver meta-parameters; e.g. kernel type, kernel parameters, acquisition function, initial variance, etc. To properly compare solvers, meta-parameters for BO and CMA-ES are selected to ensure consistent convergence without being overly greedy, and are maintained for all experiments.

4.4 Task Compatibility Optimization

In this section, the algorithm for optimizing task compatibility is developed. A set of tasks with various priorities and gains that are fixed is assumed to be given. Initial trajectories for these tasks are provided by a human operator manually selecting waypoints. Given the task

Algorithm 1 Task Compatibility Optimization

Given a set of tasks to execute on a whole-body controller.
Select task waypoint to be optimized: $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\lambda}_i$.
$\Theta = [\,]$ and $\boldsymbol{j} = [\,]$.
do
    Generate task trajectory with $\boldsymbol{\lambda}_i = \boldsymbol{\theta}_{\text{new}}$.
    Simulate task set execution.
    Compute j_c with (5).
    if first iteration **then**
        $j_c' = j_c$
    end if
    Calculate $j_c^{\text{new}} = \frac{j_c}{j_c'}$.
    $\Theta = [\Theta, \boldsymbol{\theta}_{\text{new}}]$
    $\boldsymbol{j} = [\boldsymbol{j}, j_c^{\text{new}}]$
    Update solver with Θ and $\boldsymbol{j}$.
    Get new waypoint to test $\boldsymbol{\theta}_{\text{new}}$.
while (7) is false
return $\boldsymbol{\theta}_{\text{best}}$

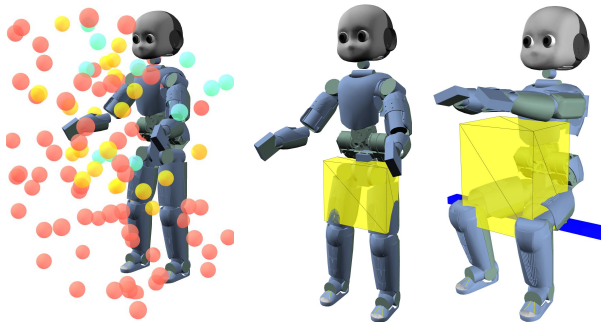
parameterization described in Section 4.1, one or more task waypoints can be used as the objective variable, $\boldsymbol{\theta}$. By modifying these waypoints, the resulting task trajectory is modified and consequently, the overall whole-body behavior. The first step in the optimization is then to select the waypoint(s) which serve as the objective variable. Here, the use of one waypoint only is considered so the objective variable is simply $\boldsymbol{\theta} = \boldsymbol{\lambda}_i$. The cost associated with $\boldsymbol{\theta}$ is then evaluated using (5) after simulating the execution of the tasks. The simulation is stopped when either all tasks have been completed or a fixed amount of time has elapsed. This initial compatibility cost is used as the baseline, j_c', to which all subsequent costs are compared. The observed parameter and cost samples are concatenated together into Θ and $\boldsymbol{j}$, respectively, and provided to either BO or CMA-ES. The solver proposes a new objective variable to test, $\boldsymbol{\theta}_{\text{new}}$ to use as the task waypoint and the task trajectory is regenerated. The task set is then executed and its compatibility cost, j_c^{new}, is calculated. Both $\boldsymbol{\theta}_{\text{new}}$ and j_c^{new} are concatenated to the current objective variable-cost pairs, and the solver is updated again. This process is iterated until,

$$\|\boldsymbol{\theta}_{\text{new}} - \boldsymbol{\theta}_{\text{best}}\| \leq \Psi \ , \tag{7}$$

where $\boldsymbol{\theta}_{\text{best}}$ is the best observed waypoint with the lowest cost and Ψ is a meta-parameter which dictates the minimum threshold for convergence. When converged, the algorithm returns the optimal waypoint, $\boldsymbol{\theta}_{\text{best}}$, with the best observed compatibility cost. This is detailed in Algorithm 1.

5 Experiments

The experiments presented in this section are designed to illustrate the task compatibility optimization described in Section 4, compare the performances of BBO solvers used within



(a) (b) (c)

Figure 2: Figure (a) shows the random reaching targets used to provide a statistical analysis of the task compatibility optimization described in Section 4. The target spheres are color coded to indicate their test case, with green meaning reachable, orange meaning possibly reachable, and red meaning unreachable. These test cases are detailed in Section 5.1. Figures (b) and (c) show the $\boldsymbol{\theta}_{\text{new}}$ bounding boxes used for the task compatibility optimization of the reaching and standing experiments, respectively.

the method, and highlight the subtle complexities of common tasks from the perspective of whole-body control. A simulation of a humanoid robot, iCub, is used for these studies. Gazebo is used as the simulation environment with the ODE physics engine.

The first experiment explores basic *reaching* movements under bipedal equilibrium, and serves as a benchmark for the task compatibility optimization method. It provides us with useful statistics to analyze the method and BBO solvers. The second experiment, entitled *standing* presents a more dynamically complex scenario in which the robot starts from a seated position and must transition to standing. Here, the difficulties of contact transitioning and dynamic equilibrium is studied. In both experiments, balance is achieved by keeping the robot's CoM position over its Polygon of Support (PoS). The PoS is defined by the convex hull of the active model contacts in the whole-body controller.

Each experiment begins with a fixed set of tasks, parameters and gains. In the following, right hand and CoM tasks only are discussed, but it should be noted that postural, torso orientation, and left hand tasks are also active during the motion. The tasks all have initial trajectories which are generated from waypoints, picked by an expert operator. For both reaching and standing the CoM task trajectory is optimized to improve the task compatibility cost. Box constraints for the optimization can be applied in both BO and CMA-ES and are set here using static stability constraints, i.e. the projection of the CoM must remain inside the PoS. The z bounds are chosen as $0.3\text{m} \leq z \leq 0.52\text{m}$. The bounding boxes for both experiments are shown in Figs. 2b and 2c.

All code for these experiments is open-source and can be found here: `https://github.com/rlober/ra-l_2017.git`.

5.1 Reaching

This experiment is concerned with demonstrating the flexibility of the proposed task compatibility optimization, as well as gleaning useful statistics about the two proposed solvers, BO

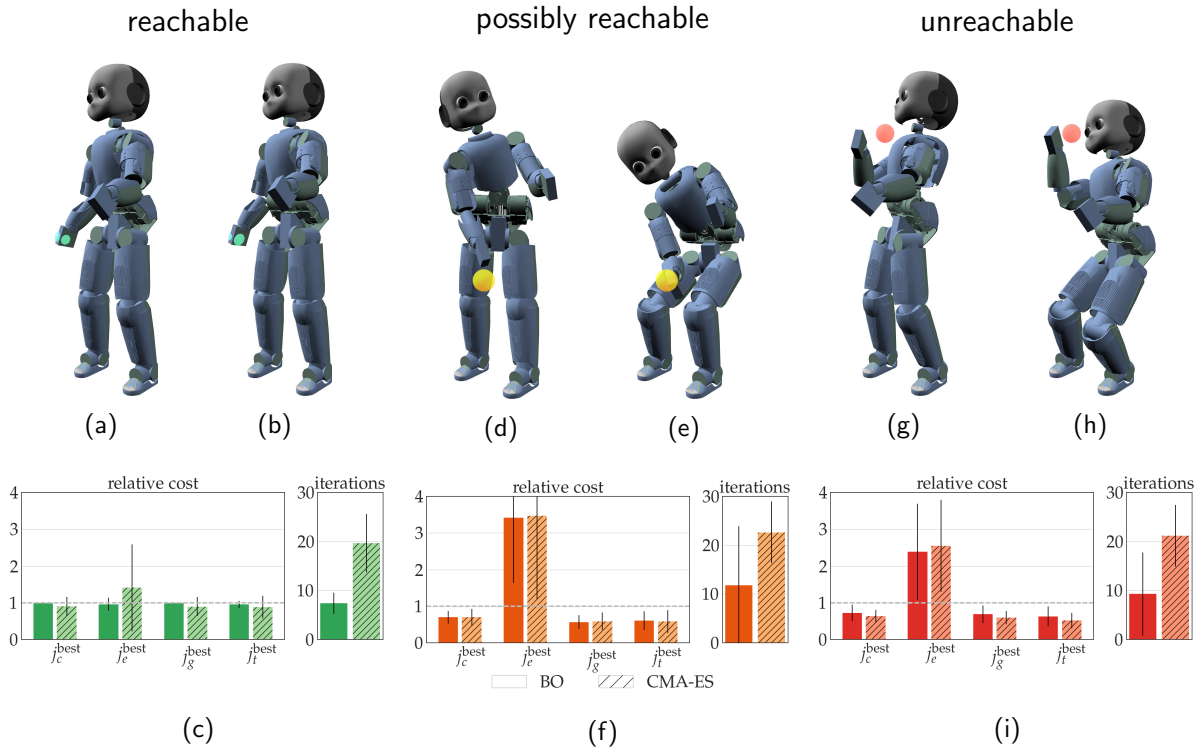


Figure 3: Results of 100 reaching experiments used to study the task compatibility optimization method. An average example is presented for the three possible reach cases. In the reachable case, (a) & (b), both the original and optimized movements attain the reach target. In the possibly reachable case, (d) & (e), the original movement does not attain the target but the optimized movement does. Finally in the unreachable case, (g) & (h), neither movement attains the target, but the optimized movement reduces the target error. For each case, the relative cost means and standard deviations are plotted for both the BO and CMA-ES solvers. Any relative cost lower than 1.0 is an improvement (the 1.0 line is indicated by a dashed grey lines). For all three cases the relative compatibility cost j_c^{best} is always less than or equal to 1.0. This indicates that the optimized movements will always be as good if not better than the original movements. In the reachable case, (c), little improvement is seen because the tasks are already compatible. For the possibly reachable and unreachable cases, (f) and (i), the compatibility is improved by reducing the tracking and goal costs at the expense of increased energy usage. For each of the cases BO outperformed CMA-ES in convergence iterations on average, but was less consistent across-the-board. See Section 6.1 for more details.

and CMA-ES. To accomplish this, 100 reach targets are randomly generated around the simulated robot, see Fig. 2a. For each target, a straight line trajectory is generated for the right hand task between its starting position and the target's position. The CoM task trajectory is generated between two waypoints,

$$\Lambda_{\text{CoM}} = \begin{bmatrix} \lambda_{\text{start}} & \lambda_{\text{goal}} \end{bmatrix} \ , \tag{8}$$

where λ_{start} is the initial CoM position, and λ_{goal} is the desired goal CoM position, which is initially chosen to be the center of the PoS at the current CoM z height. λ_{goal} is chosen as the task compatibility objective variable, $\boldsymbol{\theta}_{\text{new}}$. By optimizing λ_{goal} the CoM trajectory is modified. The compatibility cost, (5), is computed using right hand and CoM tasks.

The objective of the experiment is to attain the reach target and a target is considered attained when the right hand task frame is within 3.0cm of it. The movement is stopped if the target is attained or if a time limit is exceeded. For each reach target, the optimization is run using both BO and CMA-ES as solvers. The target is then classified into one of three cases. If the robot attains the target with the original CoM trajectory, the target is considered **reachable**. If it is unable to attain the target with the original CoM trajectory, but attains the target with the optimized CoM trajectory then the target is **possibly reachable**. Finally, if the reach target is unattainable with either the original or optimized CoM trajectories, then it is considered **unreachable**.

5.2 Standing

In this experiment, the robot is seated on a stationary bench and the objective is to stand up. In this case the only primary task is the CoM task, and its trajectory is defined by three waypoints,

$$\Lambda_{\text{CoM}} = \begin{bmatrix} \lambda_{\text{start}} & \lambda_{\text{middle}} & \lambda_{\text{goal}} \end{bmatrix} , \tag{9}$$

where λ_{start} and λ_{goal} have the same meaning as in (8) and λ_{middle} is a middle waypoint between the start and goal CoM positions. Here, λ_{goal} is picked as a nominal standing CoM position near the middle of the PoS in x and y and at approximately 0.5m from the ground, and λ_{middle} is picked as a point halfway between λ_{start} and λ_{goal}. In this experiment, λ_{middle} is chosen as $\boldsymbol{\theta}_{\text{new}}$.

In order to stand, the bench contacts used in the whole-body controller model must be deactivated or the robot will never be able to get up. The bench contacts are deactivated arbitrarily at 2.0 seconds. The motion is executed until the CoM task has attained λ_{goal} or some time limit has been exceeded. The compatibility cost, (5), is computed using only the CoM task.

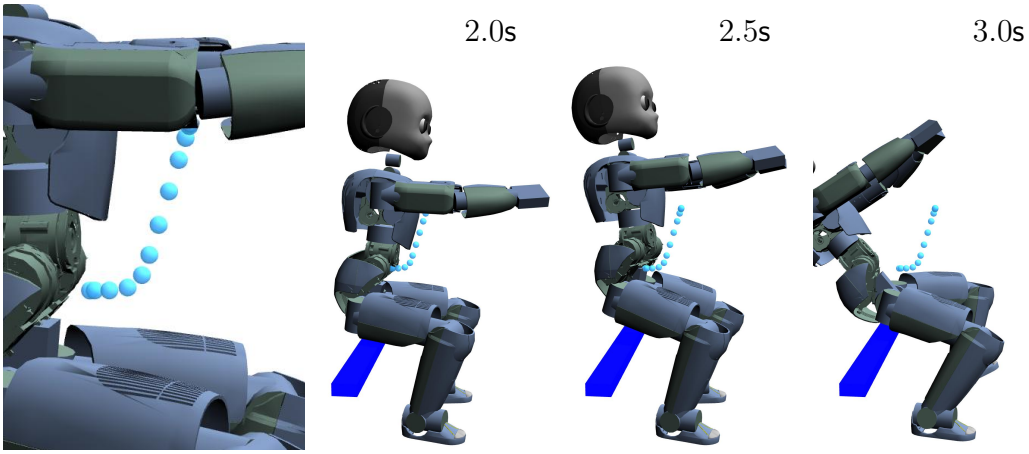
6 Results

In this section the results for the reaching and standing experiments are presented. Please see the accompanying video for a better look at the results presented here.

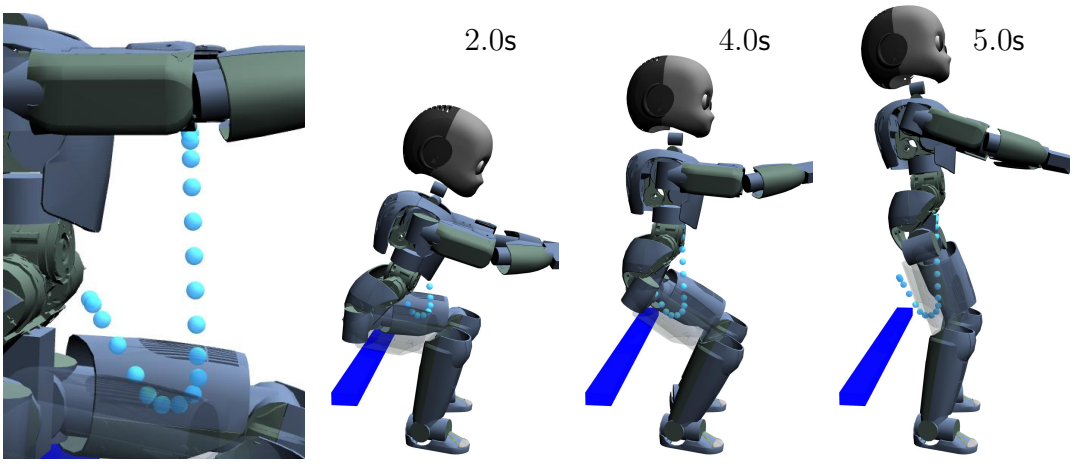
6.1 Reaching

In Fig. 3 shows representative examples of the three reach cases. For each, the relative cost means and optimization iteration means are computed. The relative compatibility cost, j_c^{best}, and the component relative costs, j_e^{best}, j_g^{best}, and j_t^{best}, are calculated by dividing the optimized costs by the original costs.

In the reachable case, Figs. 3a, 3b and 3c, it can be seen that the original task trajectories go unmodified because they are already compatible. In a few cases CMA-ES is able to improve slightly on the compatibility cost. Compatibility cost improvements can be observed between 30%-50% in the possibly reachable case, Figs. 3d, 3e and 3f, where both solvers



(a) original



(b) optimized

Figure 4: Original and optimized CoM reference trajectories and their resultant whole-body motions. The original trajectory produces an unstable standing motion causing the robot to lose balance. The optimized CoM trajectory, however, produces a successful sit-to-stand transition. The right hip is translucent in (b) to make the reference trajectory visible.

quickly converge on solutions which reduce the tracking and goal errors allowing the robot to attain the reach target. These improvements require increased energy usage to move the CoM, but the resulting successful reach which finishes more quickly, amortizing the impact of the increased energy cost. In the unreachable case, Figs. 3g, 3h and 3i, cost improvements similar to those in the possibly reachable case can be seen, despite the fact that the targets are physically unattainable.

For all three cases, BO and CMA-ES show similar performance in terms of cost reduction. In terms of convergence iterations, BO tends to find an optimum in approximately half the number of iterations needed by CMA-ES, however is less consistent than CMA-ES across-the-board.

6.2 Standing

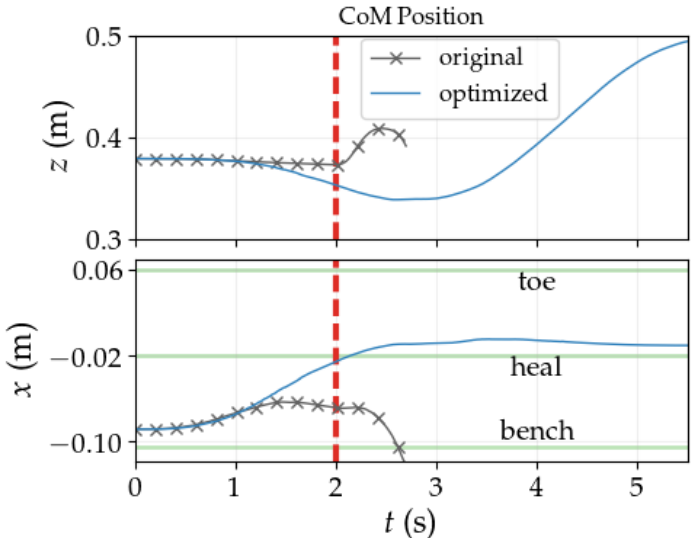


Figure 5: Evolution of the CoM for the original and optimized movements. The original CoM curves are cut off after 2.7 seconds when the robot loses balance. The red dashed line indicates the moment when the bench contacts are deactivated in the whole-body controller.

In Fig. 4, the evolution the CoM for the original and optimized movements is provided. The whole-body motion produced by the original CoM trajectory, Fig. 4a, is unstable and causes the robot to loose balance. The optimized CoM trajectory, on the other hand, produces a stable sit-to-stand transition as shown in Fig. 4b. In Fig. 5 it is observed that, at the moment the bench contacts are deactivated in the controller (the dashed vertical red line), the original motion immediately tends to lift the CoM upwards, despite an inappropriate x-location of the CoM (not close enough to the foot PoS). This inconsistent CoM trajectory does not respect the dynamic balancing conditions (see [11]) and causes the robot to fall.The optimized trajectory moves the CoM more aggressively in the forward direction as well as lowering it prior to the contact deactivation instant. The resulting CoM trajectory is balance consistent, thus leading to a successful sit-to-stand transition.

A video associated to these results and submitted with **[4]** can be viewed here: `https://tinyurl.com/hszx2qs`.

6.3 Towards year 4 demonstration

In preparation of the final year demonstration, the COM task optimization for standing is also tested in simulations including external support. Here external support is simulated, in simple and naive way, as an external force applied at each forearm of the robot through out the overall movement. The magnitude of the force at each forearm is chosen small: $F_{support} = 5.23\ N$ pointing $45°$ upward and front in the sagittal plane of the robot. Including the two aforementioned "original" and "optimized movement", two extra cases are tested: original movement with external support and optimized movement with external support. The resulting COM trajectories are presented in Fig. 6.

The results in terms of compatibility cost optimization are presented in Fig. 7. As intuitively expected, the use of an external supporting force together with an optimized COM trajectory

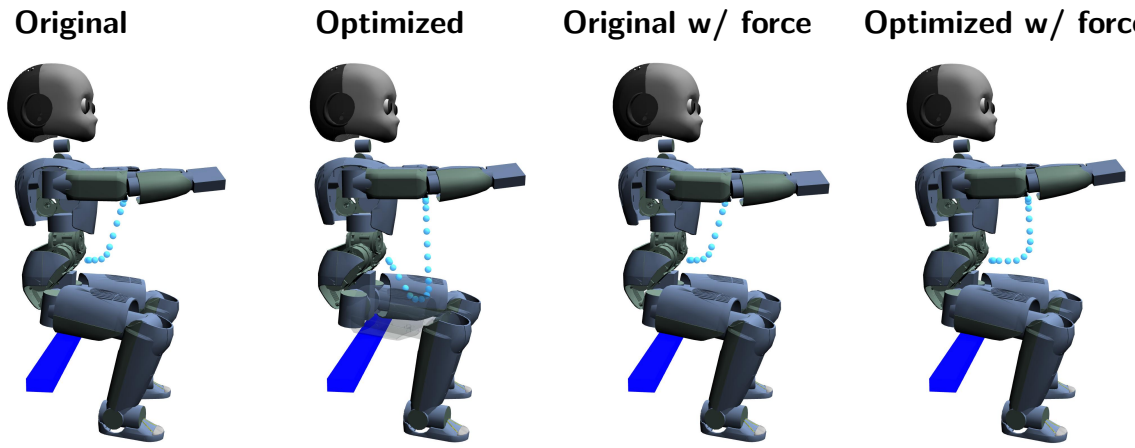


Figure 6: Original and optimized CoM reference trajectories. The original trajectory produces an unstable standing motion causing the robot to lose balance. The optimized without support and externally supported cases produce a successful sit-to-stand transition.

yields the best results in terms of tracking and whole-body energy expenditure. The main difference between the optimized motion without external support and the optimized motion with external support lies in the energetic expenditure and time needed to reach a standing posture. This is also illustrated in Fig. 8 where the time needed to reach the COM height $z_{COM} = 0.5\ m$ is more than $1.5\times$ larger (measured from the instant where bench contact constraints are no longer enforced) in the case where no external support is present.

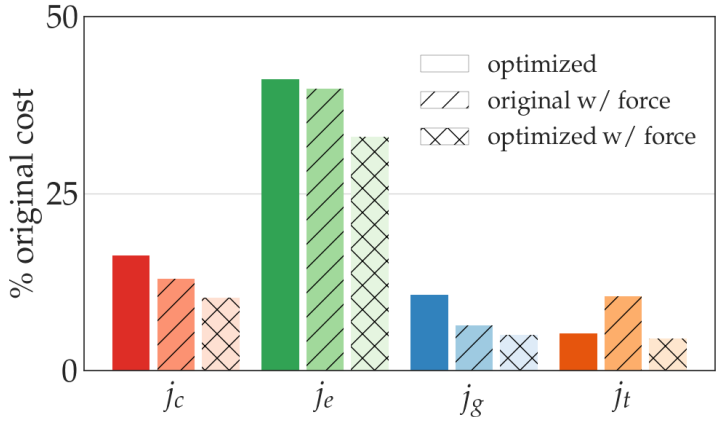


Figure 7: The costs presented here corresponds to the three following cases: optimized movement without external support, original movement with external support and optimized movement with external support. Each cost is presented as a ratio of the original cost (no optimization, no external support). As intuitively expected, the use of an external supporting force together with an optimized COM trajectory yields the best results in terms of tracking and whole-body energy expenditure.

7 Conclusion

The primary conclusion to draw from this work is that through task compatibility optimization, the overall performance of a wide variety of movements can be improved using generic principles. By improve, it is meant that the original intent of the planned task trajectories

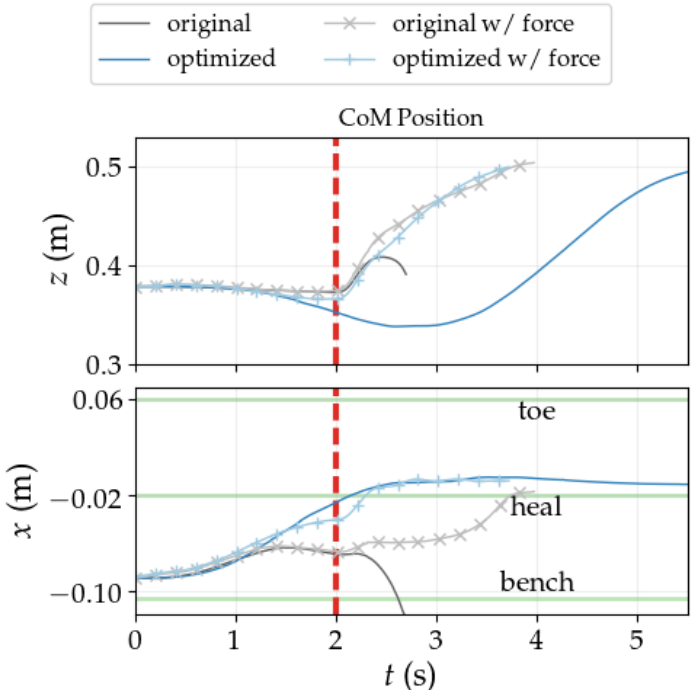
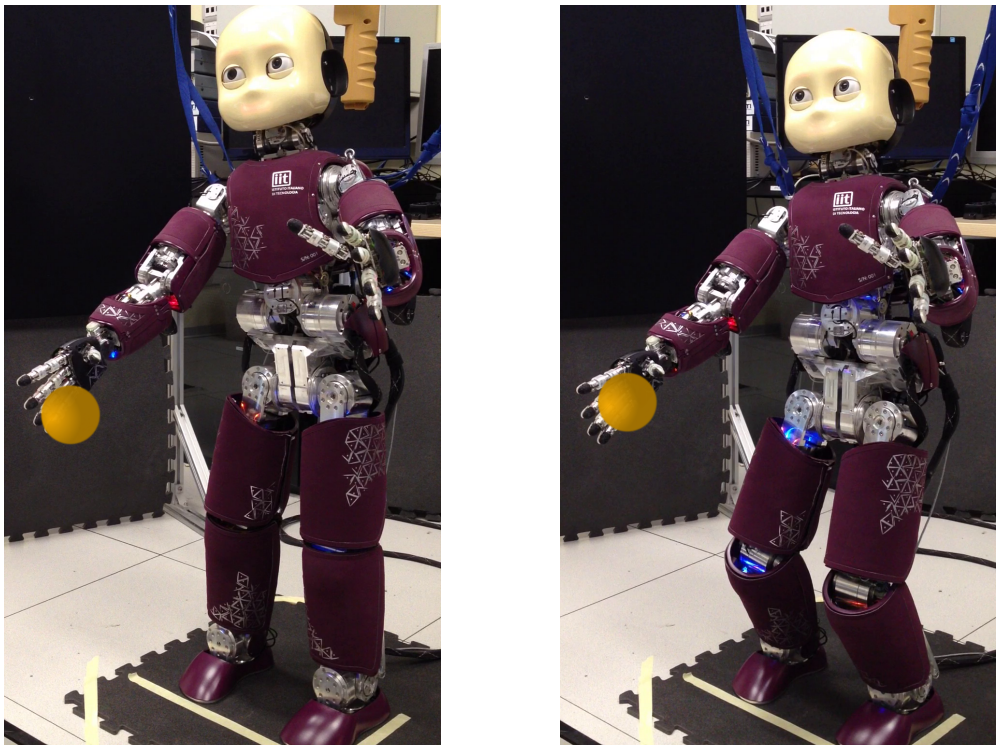


Figure 8: Evolution of the CoM for the original and optimized movements, without and with support. The original CoM curves without support are cut off after 2.7 seconds when the robot loses balance. The red dashed line indicates the moment when the bench contacts are deactivated in the whole-body controller.

is better realized. In the reaching experiments this means more targets are attained, and in the standing case it means the robot can successfully transition from sitting to standing. Moreover, the need for fine tuning of controller priorities and gains is alleviated because they are accounted for in the task optimization. In addition, the compatibility optimization concept is controller independent and, while it is expected that the controller described in section 2 will be used during the final year demo, the use of another controller would not imply any modification at the task compatibility optimization level.

With regards to BBO solvers, a comparison of the BO and CMA-ES reveals that BO tends to converge to an optimal solution in half the number of simulation iterations needed by CMA-ES. Such efficiency is desirable if the optimization iterations are to occur on a real robot.

Finally, it is believed that the proposed task compatibility optimization loop is an important first step towards an intermediate control layer between high-level planning and low-level control, where task trajectories are optimized for compatibility before being executed on the real robot, as illustrated by Fig. 9 for a reaching motion. The combination of state-of-the-art whole-body control and task level optimization will hopefully also be demonstrated during the final year demonstration. Improved results can even be expected using the estimation of the supporting wrenches applied by the caregiver developed in Deliverable 5.4 [34].



(a) original (b) optimized

Figure 9: An original and optimized reaching motion executed on an iCub robot is illustrated here. These preliminary results show that the movements produced by task compatibility optimization may be viable on real platforms.

References

[1] M. Azzad, M. Liu, M. Mistry, F. Nori, V. Padois, D. Pucci, F. Romano, and S. Traversaro, "Codyco project, deliverable 3.2: Local solver in compliant-world cases," Sorbonne Universités, UPMC Paris 06, UMR CNRS 7222, Tech. Rep., Feb. 2016.

[2] D. Pucci, F. Romano, J. Eljaik, S. Traversaro, S. Ivaldi, V. Padois, and F. Nori, "Codyco project, deliverable 5.3: Validation scenario 3: balancing on compliant environmental contacts," Italian Institute of Technology & Sorbonne Universités, UPMC Paris 06, UMR CNRS 7222, Tech. Rep., Feb. 2016.

[3] R. Lober, V. Padois, O. Sigaud, V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, S. Ivaldi, and A. Paraschos, "Codyco project, deliverable 4.3: Learning of the prioritization policies," Technische Universität Darmstadt & Jozef Stefan Institute, Tech. Rep., Feb. 2016.

[4] R. Lober, V. Padois, and O. Sigaud, "Task compatibility optimization," *IEEE Robotics and Automation Letters*, 2017, submitted.

[5] 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. 01, pp. 29–43, 2004.

[6] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 1283–1290.

[7] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Soueres, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, 2013.

[8] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4414–4419.

[9] A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from Mixed-Integer Model Predictive Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, USA, Sept 2014, pp. 4014 – 4021.

[10] J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2015, pp. 3346–3351.

[11] N. Perrin, D. Lau, and V. Padois, "Effective Generation of Dynamically Balanced Locomotion with Multiple Non-coplanar Contacts," in *International Symposium on Robotics Research*, 2015.

[12] K. Bouyarmane and A. Kheddar, "Humanoid robot locomotion and manipulation step planning," *Advanced Robotics*, vol. 26, no. 10, pp. 1099–1126, 2012.

[13] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, Dec 2014.

[14] V. Padois, "Control and design of robots with tasks and constraints in mind," Habilitation Diriger des Recherches, Universit Pierre et Marie Curie, Paris, France, Oct. 2016. [Online]. Available: http://hal.archives-ouvertes.fr/tel-01398868/en

[15] A. Ibanez, P. Bidaud, and V. Padois, "Optimization-based control approaches to humanoid balancing," in *Humanoid Robotics: a Reference*, A. Goswami and P. Vadakkepat, Eds. Springer Netherlands, 2016, accepted for publication.

[16] K. Bouyarmane and A. Kheddar, "On Weight-Prioritized Multi-Task Control of Humanoid Robots," *IEEE Transactions on Automatic Control*, in revision 2015.

[17] P.-B. Wieber, A. Escande, D. Dimitrov, and A. Sherikov, "Geometric and numerical aspects of redundancy," in *Geometric and Numerical Foundations of Movements*, 2017.

[18] 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, no. 7, pp. 1006–1028, 2014.

[19] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1385–1400, 2015.

[20] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, vol. 40, no. 1, pp. 17–31, 2016.

[21] M. Liu, R. Lober, and V. Padois, "Whole-body hierarchical motion and force control for humanoid robots," *Autonomous Robots*, vol. 40, no. 3, pp. 493–504, 2016.

[22] R. Lober, V. Padois, and O. Sigaud, "Variance modulated task prioritization in Whole-Body Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2015, pp. 3944–3949.

[23] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi, "Learning soft task priorities for control of redundant robots," in *IEEE International Conference on Robotics and Automation*, May 2016, pp. 221–226.

[24] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi, "Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, 2016, pp. 101–108.

[25] A. Paraschos, J. Peters, and G. Neumann, "Probabilistic prioritization of movement primitives," 2017, under review.

[26] R. Lober, V. Padois, and O. Sigaud, "Multiple task optimization using dynamical movement primitives for whole-body reactive control," in *IEEE-RAS International Conference on Humanoid Robots*, Nov 2014, pp. 193–198.

[27] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[28] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," 2012.

[29] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen, "Information-geometric optimization algorithms: A unifying picture via invariance principles," *arXiv preprint arXiv:1106.3708*, 2011.

[30] C. E. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[31] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian gait optimization for bipedal locomotion," in *International Conference on Learning and Intelligent Optimization*. Springer, 2014, pp. 274–290.

[32] R. Antonova, A. Rai, and C. G. Atkeson, "Sample efficient optimization for learning controllers for bipedal locomotion," in *IEEE-RAS International Conference on Humanoid Robots*, Nov 2016, pp. 22–28.

[33] P. Englert and M. Toussaint, "Combined Optimization and Reinforcement Learning for Manipulations Skills," in *Robotics: Science and Systems*, 2016.

[34] D. Pucci, F. Romano, J. Eljaik, S. Traversaro, S. Ivaldi, V. Padois, and F. Nori, "Codyco project, deliverable 5.3: Validation scenario 3: balancing on compliant environmental contacts," Italian Institute of Technology & Sorbonne Universités, UPMC Paris 06, UMR CNRS 7222, Tech. Rep., Feb. 2016.