Learning robust task priorities of QP-based whole-body torque-controllers

Marie Charbonneau[1,2], Valerio Modugno[2,3], Francesco Nori[4], Giuseppe Oriolo[3], Daniele Pucci[1] and Serena Ivaldi[2]

***Abstract*— Generating complex whole-body movements for humanoid robots is now most often achieved with multi-task whole-body controllers based on quadratic programming. To perform on the real robot, such controllers often require a human expert to tune or optimize the many parameters of the controller related to the tasks and to the specific robot, which is generally reported as a tedious and time consuming procedure. This problem can be tackled by automatically optimizing some parameters such as task priorities or task trajectories, while ensuring constraints satisfaction, through simulation. However, this does not guarantee that parameters optimized in simulation will also be optimal for the real robot. As a solution, the present paper focuses on optimizing task priorities in a robust way, by looking for solutions which achieve desired tasks under a variety of conditions and perturbations. This approach, which can be referred to as domain randomization, can greatly facilitate the transfer of optimized solutions from simulation to a real robot. The proposed method is demonstrated using a simulation of the humanoid robot iCub for a whole-body stepping task.**

I. Introduction

Applications involving highly redundant robots, such as humanoids, have the potential to disrupt many industries and bring countless benefits to society. Nevertheless, the design of efficient and safe controllers for humanoid platforms is highly challenging, especially for applications involving interaction with the environment. Indeed, a great number of issues can stem from the complexity of the control algorithms, task inconsistencies, or the need of guaranteeing constraints satisfaction, in order to generate physically consistent behaviors. A highly effective solution to the whole-body control problem is to decompose a complex behavior into several elementary tasks, according to a prioritization scheme [1], typically either a "strict" or a "soft" task hierarchy.

With strict prioritization strategies (such as a traditional *stack-of-tasks*), a fixed task hierarchy is assured by geometrical conditions such as a null space task projector [2], [3], or by the use of quadratic programming (QP) to compute optimal control actions [4]. Conversely, prioritization in a soft task hierarchy (based on a weighted combination of tasks), can be achieved by assigning each task a weight defining its relative importance [5], [6].

This work was supported by the EU H2020 program under the Marie Sklodowska-Curie SECURE grant (n.642667), as well as the european projects An.Dy (n.731540) and Comanoid (n.645097).

[1]iCub Facility, Istituto Italiano di Tecnologia, Genova, Italy. name.surname@iit.it

[2]Loria, Inria Nancy - Grand Est, CNRS & Univ. Lorraine, Villers-lès-Nancy, France. name.surname@inria.fr

[3] DIAG, Sapienza Universita di Roma, Roma, Italy. surname@diag.uniroma1.it

[4]Google DeepMind, London, UK. fnori@google.com

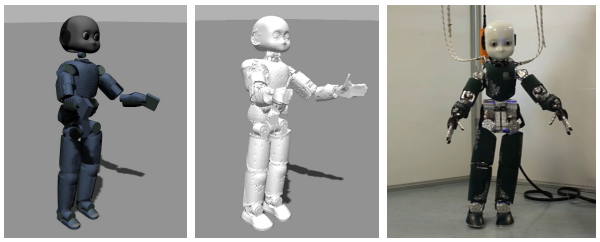


Fig. 1: Different iCub models performing the same whole-body motion with several tasks. In this paper, with the purpose to eventually ease the passage from simulated to real-world robots, we optimize the task priorities for robustness, in order to allow their transfer from one robot model to a different one, without the need of re-tuning.

Usually, task priorities are designed *a priori* by experts, then manually tuned to adjust the task ordering, timing, transitions, *etc*. It is a tedious operation which, depending on the complexity of the system and the task, can take a notable portion of a researcher's time, and can be particularly hard in the case of whole-body control of floating-base platforms.

A recent line of research seeks to tackle this issue, aiming to automatically learn whole-body task priorities [7], [8], [9]. For instance in [7], task coefficients are learned for controlling a single arm, allowing the transfer of acquired knowledge to different tasks. In [8], task generalization is achieved with an evolutionary algorithm employed to learn policies for a range of tasks using parametrized motor skills. Then, in [9], task priorities guaranteeing the non violation of system constraints are learned for bimanual motions.

Learning methods often use a random exploration over numerous experiments, which could be problematic to perform on hardware. For this reason, training is preferably performed in simulation. However, inherent differences between simulated and real robots can render an optimal solution *untransferrable* from one to the other. This gap between reality and simulation needs to be accounted for, in order to achieve automatic parameter tuning.

Solutions to this problem have recently been addressed by trial-and-error algorithms [10], [11]. In [10], prior knowledge from simulations was exploited to find acceptable behaviors on the real robot, in few trials. In [11], a trial-and-error learning algorithm encouraged exploration of the task space, allowing adaptation to inaccurate models, also in few trials.

Learning from simulation can also simplify data gathering, but there is no guarantee that the acquired knowledge is effective in the real world. For example, optimal control and

movement primitives are combined in [12] to find solutions which can be easily deployed on the real robot, but they strongly rely on the accuracy of the simulated model.

Also, since QP solvers often allow for constraints relaxation, strict constraints satisfaction is not always ensured by the frameworks presented above. In [9], the learning procedure guarantees strict constraints fulfillment, but transferring knowledge from simulation to reality did not fully achieve the desired behavior. This implies that constraints satisfaction is beneficial, but not enough to achieve transferability: solutions which are *robust* rather than *optimal* are needed, in order to achieve a better generalization.

Looking for robust solutions is central to *transfer learning*, which seeks to exploit previously acquired knowledge [13] to allow a model trained on one task to be re-purposed on a second related one. This approach can benefit from *domain randomization* (DR) [14], which consists in randomizing some aspects of the simulation to enrich the range of possible environments experienced by the learner. In [15], robust policies for pivoting a tool held in the robot's gripper were learned in simulation, given random friction and control delays, such that the learned policies proved to be effective on the real robot as well.

This work proposes to apply the idea of DR to whole-body controllers. In this context, we want to ensure that balance is maintained while performing a task, even if large differences exist between the learning domain and the testing domain.

To achieve this result, we combine a DR approach with fitness functions promoting the robustness of the learned controller, and provide a method to learn robust task priorities which achieve desired goals, while allowing to facilitate the transfer of results from simulation to reality. The effectiveness of the proposed method is demonstrated by optimizing parameters in simulation for the goal of making the iCub perform steps, and showing that it is possible to overcome issues related to the transferability problem.

The paper is organized as follows. Section II introduces the modeling of a floating-base robot for whole-body control. Section III describes the task-based whole-body controller used for validating the approach, as well as the constrained optimization algorithm used for learning task priorities. Experiments led with different models of the iCub robot are then illustrated in Section IV. Results are discussed in V, leading to the conclusion of the paper.

II. Background

A. Notation used throughout the paper

- The set of real numbers is denoted by $\mathbb{R}$
- The transpose operator is denoted by $(\cdot)^\top$
- The Moore-Penrose pseudoinverse is denoted by $(\cdot)^\dagger$
- $|v|$ denotes the Euclidean norm of a vector $v \in \mathbb{R}^n$
- Given a time function $f(t) \in \mathbb{R}^n$, its first- and second-order time derivatives are denoted by $\dot{f}(t)$ and $\ddot{f}(t)$

B. System Modelling

The robot is modelled as described in [3], where it is assumed to be a free-floating system composed of $n+1$ links

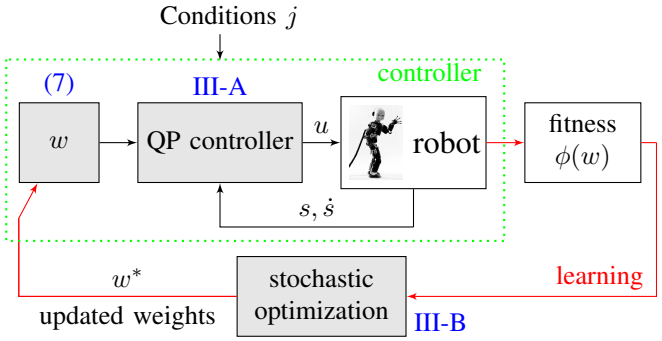


Fig. 2: Overview of the proposed method. Given task priorities w, the QP-based controller computes a control input u under a set of conditions j (e.g. desired trajectories, disturbances, noise). An outer learning loop allows the optimization of the task weights.

connected with n 1-DOF (degree of freedom) joints. The robot configuration space is characterized by q, composed of the position and orientation of a base frame $\mathcal{B}$ attached to a link of the robot, and the joint angles s. The robot velocity is expressed as $\nu = (\mathrm{v}_\mathcal{B}, \dot{s})$, where $\mathrm{v}_\mathcal{B}$ is the velocity of the base frame expressed with respect to the inertial frame.

Applying the Euler-Poincaré formalism [16, Chapter 13.5] yields the following system equation of motion:

$$M(q)\dot{\nu} + h(q,\nu) = \zeta\tau + \sum_{k=1}^{n_c} J_{c_k}^\top f_k \tag{1}$$

where $M \in \mathbb{R}^{n+6\times n+6}$ is the mass mass matrix, $h \in \mathbb{R}^{n+6}$ is the bias vector of Coriolis and gravity terms, $\tau \in \mathbb{R}^n$ is a vector representing the actuation joint torques, and $\zeta = (0_{n\times 6}, 1_n)^\top$ is a selector matrix, in which $1_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix and $0_{n\times 6} \in \mathbb{R}^{n\times 6}$ is a zero matrix. Assuming that the robot is interacting with the environment through n_c distinct wrenches (as an abuse of notation, here we do not define *wrench* as the dual of a *twist*), $f_k \in \mathbb{R}^6$ denotes the k-th external wrench applied by the environment on the frame c_k of the robot.

III. Methods

The method proposed for learning robust task priorities (outlined in Fig. 2) relies on two main parts: (i) a QP-based whole-body torque-controller which tracks desired task trajectories and sends joint torque commands to the robot, and (ii) a black-box constrained stochastic optimization procedure, posing no restriction on the structure of the learning problem. It is used to optimize task priorities w as follows: at the end of a roll-out (i.e. execution of a footstep), the fitness of the obtained robot behavior is evaluated, allowing the optimization algorithm to update the task weights.

A. QP-based whole-body torque controller

The control framework used in this paper was defined from a stack of tasks solved by quadratic programming. The following subsections define the control input and objective, before describing an optimization problem allowing the computation of a control input which achieves the objective.

1) Control input: A control input $u = \begin{bmatrix}\tau^\top & F_c^\top\end{bmatrix}^\top$ is composed of joint torques τ and contact forces F_c, where $F_c \in \mathbb{R}^{6n_c}$ is a stacked vector of contact wrenches. Defining J_c as the contact Jacobian, stacked in the same way as F_c, and $B = \begin{bmatrix}\zeta & J_c^\top\end{bmatrix}$, the system dynamics (1) can then be reformulated as

$$M(q)\dot{\nu} + h = Bu \tag{2}$$

2) Control objective: The developed controller has the following objectives, from which a stack of tasks is defined:

- Stabilize the center of mass position $X_{CoM} \in \mathbb{R}^3$
- Stabilize the stance foot pose $X_{stance} \in \mathbb{R}^6$
- Stabilize the swing foot pose $X_{swing} \in \mathbb{R}^6$
- Stabilize the neck orientation $X_{neck} \in \mathbb{R}^3$
- Track joint positions (postural task) s
- Minimize joint torques τ

The velocity $\dot{X}_T$ of a Cartesian task T can be computed from the associated Jacobian J_T and the robot velocity ν with $\dot{X}_T = J_T\nu$, where $T \in \{CoM, stance, swing, neck\}$. Deriving this expression yields the task acceleration:

$$\ddot{X}_T = \dot{J}_T\nu + J_T\dot{\nu} \tag{3}$$

In view of Eq. (2) and (3), task accelerations $\ddot{X}_T$ can be formulated as a function of the control input u:

$$\ddot{X}_T(u) = \dot{J}_T\nu + J_T M^{-1}(Bu - h) \tag{4}$$

Stabilization of a Cartesian task may then be attempted by minimizing $\tilde{\ddot{X}}_T(u) = \ddot{X}_T(u) - \ddot{X}_T^*$, the error on the task acceleration, where $\ddot{X}_T^*$ is a feedback term composed of reference linear accelerations obtained with a proportional-derivative (PD) feedback control policy, and reference angular accelerations computed as in [17, section 5.11.6, p.173].

As for the postural task, similarly as above, in view of Eq. (2), one can obtain $\ddot{s}(u)$, a formulation of the joint accelerations $\ddot{s}$ as a function of the control input u:

$$\ddot{s}(u) = \zeta^\top \left[M^{-1}(Bu - h)\right] \tag{5}$$

Stabilization of the postural task may then be attempted by minimizing $\tilde{\ddot{s}}(u) = \ddot{s}(u) - \ddot{s}^{ref}$, the error on the joint accelerations, with $\ddot{s}^{ref}$ obtained through inverse kinematics by solving the following optimization problem:

$$\ddot{s}^{ref} = \arg\min_{\ddot{s}} \ \frac{1}{2}\sum_T w_T \left|\tilde{\dot{\nu}}_T\right|^2 + w_s \left|\tilde{\dot{\nu}}_s\right|^2 \tag{6}$$

In the above, the error on robot acceleration is computed with $\tilde{\dot{\nu}}_T = \dot{\nu} - J_T^\dagger \left(\dot{J}_T\nu - \ddot{X}_T^*\right)$ for Cartesian tasks, and with $\tilde{\dot{\nu}}_s = \dot{\nu} - \begin{bmatrix}\dot{\nu}_{\mathcal{B}}^{*\top} & \ddot{s}^{*\top}\end{bmatrix}^\top$ for the postural task. $\ddot{s}^*$ is a feedback term computed using a PD control policy, and $\dot{\nu}_{\mathcal{B}}^*$ is a feedback term on the base velocity used to ensure convergence of the base acceleration to zero.

3) QP formulation: In the context where tasks may have relative priorities, soft task priorities would allow to achieve a trade-off between weighted tasks. A set of task weights is defined, in order to attribute each task a priority, with

$$\boldsymbol{w} = \{w_{CoM}, w_{stance}, w_{swing}, w_{neck}, w_s, w_\tau\} \tag{7}$$

The terms $w_{CoM}, w_{stance}, w_{swing}, w_{neck} \in \mathbb{R}$ refer to weights associated to the CoM, stance foot, swing foot and neck Cartesian tasks, and $w_s, w_\tau \in \mathbb{R}$ to the weights associated to the postural task and joint torque regularization.

The control architecture described above can thus be formulated as the following optimization problem.

$$u^* = \arg\min_u \ \frac{1}{2} \text{ cost} \tag{8a}$$

$$\text{subject to } Cu \leq b \tag{8b}$$

where the constraint Eq. (8b) ensures that the contact forces remain within the associated friction cones. The cost function (8a) is computed as the weighted sum of all task objectives: $\text{cost} = \sum_T w_T \left|\tilde{\ddot{X}}_T(u)\right|^2 + w_s \left|\tilde{\ddot{s}}(u)\right|^2 + w_\tau \left|\tau(u)\right|^2$, in which $w_T \in \mathbb{R}$ refers to weights associated to Cartesian tasks. Reorganizing the terms in the cost function, one can easily verify that it has the form of a QP problem.

B. Constrained stochastic optimization

In this work, the learning problem is cast as a black-box constrained stochastic optimization. Given a fitness function $\phi(\boldsymbol{w}) : \mathbb{R}^{n_P} \to \mathbb{R}$ (with n_P the number of task weights to optimize), sets of n_{IC} inequality constraints and n_{EC} equality constraints h, g, the idea is to find an optimal solution $\boldsymbol{w}^\circ \in \boldsymbol{W} \subseteq \mathbb{R}^{n_P}$ to the problem:

$$\boldsymbol{w}^\circ = \arg\max_{\boldsymbol{w}} \phi(\boldsymbol{w}) \tag{9a}$$

subject to

$$g_i(\boldsymbol{w}) \leq 0 \quad i = 1, \ldots, n_{IC} \tag{9b}$$

$$h_i(\boldsymbol{w}) = 0 \quad i = 1, \ldots, n_{EC} \tag{9c}$$

The Covariance Matrix Adaptation - Evolution Strategy (CMA-ES), first described in [18], is a standard tool for solving black-box optimization problems. In its basic implementation, CMA-ES defines a multivariate Gaussian distribution $\mathcal{N}(\bar{\boldsymbol{w}}, \sigma^2, \boldsymbol{\Sigma})$ with mean $\bar{\boldsymbol{w}}$, step size σ^2 and covariance $\boldsymbol{\Sigma}$. However, since CMA-ES was not originally designed for handling constraints, this work employs a constrained extension of it called (1+1)CMA-ES, which uses covariance constrained adaptation (CCA) [19]. This variant was benchmarked in [9] for bimanual tasks, as well as in [20] for trajectory optimization of a whole-body movement (standing up from a chair).

A single iteration, or *generation*, of (1+1)-CMA-ES comprises several stages. First, a sample $\boldsymbol{w}_1$ is drawn according to the rule $\boldsymbol{w}_1 = \boldsymbol{w} + \sigma\mathbf{D}\mathbf{z}$, where $\mathbf{D}$ is the Cholesky factor of the covariance matrix $\boldsymbol{\Sigma} = \mathbf{D}^\top\mathbf{D}$ and $\mathbf{z}$ is a standard normal distributed vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The fitness function ϕ is evaluated for this sample, and the Gaussian distribution is updated based on the result, as described in [21]. This update also exploits the previous generations to change the covariance and step size of the search distribution. Finally, information about the successful iterations is stored in a *search path* $\mathbf{s} \in \mathbb{R}^{n_P}$. A more detailed description of the algorithm can be found in [20].

IV. EXPERIMENTS

This section exposes experiments with the iCub robot [22]. They were performed in order to validate empirically that the method described in section III is capable of generating task priorities which (i) produce robust whole-body motions, even when contacts due to physical interaction with the environment evolve in time, and (ii) can cope with imperfections in the robot model, disturbances and noise.

1) Robot setup: Experiments were conducted in simulation using the open-source robot simulator Gazebo [23]. They were performed with the iCub robot, using 23 DOF on legs, arms and torso. The robot is equipped with 6 force/torque sensors on the arms, legs and feet, allowing whole-body dynamics computations. We tested our method using two distinct models of the iCub with different inertial properties: the first one with a tethered power supply and the second one with a battery pack on the back, as shown in Fig. 1.

2) Whole-body motion: The controller described in section III-A was developed in Matlab/Simulink using WBToolbox [24]. It can be used either for a simulated or a real robot.

It is applied to the task of performing steps, i.e. performing the following sequential movements for each step: move the CoM above the stance foot, move the swing foot up by 0.025m, move the swing foot back to its initial pose, and move the CoM back to its initial position. The passage from one movement to the next takes place when Cartesian task errors are smaller than a threshold and contact forces are consistent with the desired motion, or a maximum state duration has been reached. Over the course of a step, the desired stance foot pose, neck orientation and posture remain at their initial value.

Note that over all experiments, the proportional-derivative gains used to compute reference accelerations associated to Cartesian and postural tasks are kept constant.

The next subsections describe the experiments we performed, while Table I presents them in summary form. Training and testing were performed with different robot models and stepping tasks, as well as under different randomized conditions (RC) as described in Table II.

A. Training with the tethered iCub model

In order to optimize task weights, three different fitness functions were compared. The first one, ϕ_p, favored performance on the Cartesian tasks and less deployed effort. The second one, ϕ_r, focused on robustness, by favoring solutions with smaller excursion of the ZMP position P_{ZMP} with respect to the center of the support polygon O_{SP}. The third fitness function, ϕ_{pr}, was a combination of the first two:

$$\phi_p = -\frac{1}{X_{T_{max}}} \sum_{t=0}^{t_{end}} \sum_{T} \left| \tilde{X}_T \right|^2 - \frac{0.0001}{\tau_{max}} \sum_{t=0}^{t_{end}} |\tau|^2 \quad \text{(10a)}$$

$$\phi_r = -\frac{1}{P_{ZMP_{max}}} \sum_{t=0}^{t_{end}} |P_{ZMP} - O_{SP}|^2 \quad \text{(10b)}$$

$$\phi_{pr} = \frac{1}{2}(\phi_p + \phi_r) \quad \text{(10c)}$$

$X_{T_{max}}$, τ_{max} and $P_{ZMP_{max}}$ are normalization factors. Inequality constraints to the learning problem were defined on joint position limits and torque limits: they act as an extra safety built on top of the QP controller.

The task used for the optimization is to perform 1 step, as described in section IV-.2. The simulation was limited to 10 seconds, allowing the robot to perform one step and shift its weight to the side in order to start a second one, making sure that the robot remained stable after foot touchdown.

Early termination of a simulation took place in cases where the robot fell or the QP in Eq. (8) could not be successfully solved (which may happen when the weights being tested are far from being optimal, and lead to a configuration in which the QP solver simply can not find a solution). In these cases, a penalty of -1.5 was added to the fitness in Eq. (10).

Optimized task priorities were obtained by performing 200 iterations of (1+1)-CMA-ES [19] applied to the control framework, with an exploration rate of 0.1; each learning experiment was repeated 10 times. Since task priorities as used in Eq. (6) and (8) are relative to each other, w_{CoM} was attributed a fixed value of 1. The remaining task priorities were attributed bounds as shown in the bottom of Table I. Furthermore, the weight values $w_0 = [1, 1, 1, 0.1, 1\text{e}{-3}, 1\text{e}{-4}]$ obtained by hand were verified to allow the tethered iCub model to successfully perform the desired stepping motion, and were therefore used as a starting point.

Moreover, to achieve robustness through domain randomization and enable the controller to eventually cope with real-world data, the robot was subjected to the randomized conditions 1 to 5 (see Table II) at each learning iteration. These conditions constitute the set of conditions j under which the controller had to perform. In particular, tests performed in simulation showed that applying a force of 10N on the chest for no more than 1 second was sufficient to destabilize the robot when using unoptimal weights. Thus, using such disturbances during learning can encourage the generation of robust task priorities.

An additional set of learning experiments was performed for ϕ_{pr} without using DR, to assess the contribution of DR.

Results showed that 200 iterations were sufficient to achieve strong convergence. Weights obtained with each of the fitness functions in Eq. (10) are shown in Table I.

B. Testing 1: with the tethered iCub model

In order to validate the robustness achieved with the optimized weights, each set of them were tested on the same iCub model used for training, but not submitted to randomized conditions. The testing task in this case was to achieve 6 consecutive steps. The success rates achieved with the optimized weights from each fitness function are shown in Table I, and typical trajectories achieved using each fitness function are shown in the left part of Fig. 3 and in Fig. 4.

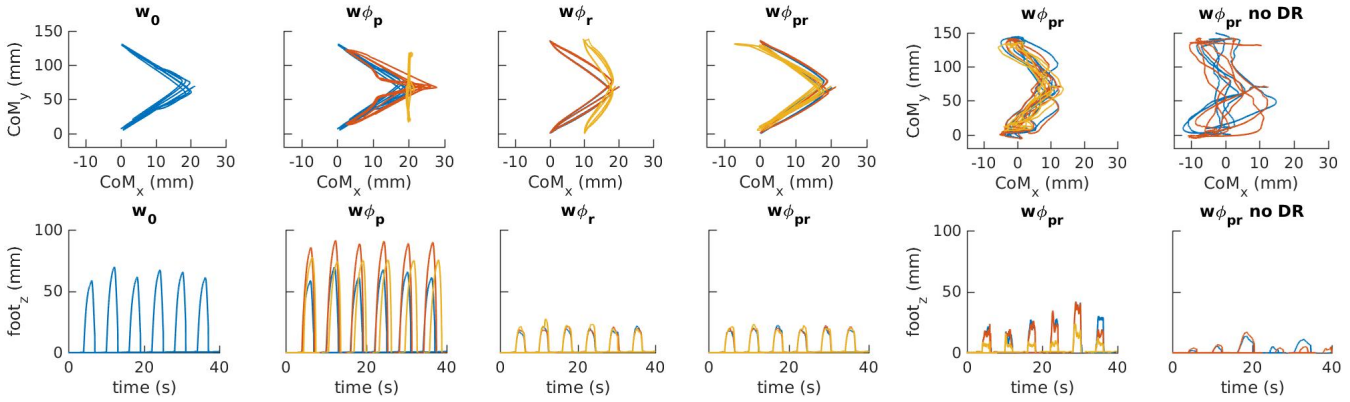
C. Testing 2: with the backpacked iCub model

In order to replicate conditions similar to performing experiments on the real robot, we prepared a second model of the iCub, with a battery pack installed on its back. This

TABLE I: Summary of performed experiments and achieved results

Scenario Robot Task RC	**Training** tethered 1 step 1, 2, 3, 4, 5								**Testing 1** tethered 6 steps —	**Testing 2** backpacked 6 steps 5, 6, 7
	Fitness	#	DR	w_{CoM}	w_{stance}	w_{swing}	w_{neck}	w_s	w_τ	success	success
	hand tuning	1	No	1	1	1	1	1e−3	1e−4	$^1/_1$	$^0/_1$
	ϕ_p	10	Yes	1	0.2 ± 0.3	1.1 ± 1.2	(1 ± 3)e−3	0.5 ± 0.3	(2 ± 5)e−5	$^5/_{10}$	$^0/_{10}$
	ϕ_r	10	Yes	1	1.6 ± 1.2	1.8 ± 1.0	0.1 ± 0.1	(4 ± 7)e−3	1e−10	$^7/_{10}$	$^5/_{10}$
	ϕpr	10	Yes	1	0.9 ± 1.3	2.4 ± 1.1	0.6 ± 1.2	1e−6	1e−10	$^{10}/_{10}$	$^{10}/_{10}$
	ϕpr	10	No	1	1.0 ± 0.3	0.1 ± 0.3	0.2 ± 0.1	1e−6	(4 ± 5)e−6	$^8/_{10}$	$^2/_{10}$
	weights lower bounds:			1	1e−4	1e−4	1e−10	1e−6	1e−10		
	weights upper bounds:			1	10	10	10	10	0.1		

Randomized conditions RC are defined in Table II. The column # gives the number of training experiments performed given each fitness function, while only 1 set of manually tuned gains was used. In the column DR, "Yes" means that the RC defined at the top of the table were used for DR while training; "No" means that they were disabled. The "success" columns report success rates achieved when testing the sets of trained weights.



(a) tethered model (b) backpacked model

Fig. 3: Typical CoM and feet trajectories obtained for 6 strides performed with the tethered and backpacked iCub models, given initial weights w_0 or weights optimized using DR (except for "no DR") with ϕ_p, ϕ_r, ϕ_{pr}. Each color denotes the use of a different set of optimized weights. The x, y and z axes correspond respectively to the sagittal, frontal and vertical axes.

TABLE II: Randomized set of conditions j

Randomized Condition (RC) in j	Random value
1. Gaussian noise on F/T sensor signals	On / Off
2. Appointed swing foot	Left / Right
3. Direction in which swing foot is moved	Front / Back
4. X_{CoM_d} moved forward by δ (m)	$\{\delta \mid \delta \in \mathbb{R}^+, \delta \leq 0.02\}$
5. n_F external wrenches applied on chest	$\{n_F \mid n_F \in \mathbb{Z}, n_F \leq 7\}$
and for each wrench i:	
at time t_{F_i} (s, with 1e−2 precision)	$\{t_{F_i} \mid t_{F_i} \in \mathbb{R}^+, t_{F_i} \leq 10\}$
duration d_F (s, with 1e−2 precision)	$\{d_{F_i} \mid d_{F_i} \in \mathbb{R}^+, d_{F_i} \leq 1\}$
direction $(\gamma_{F_i}, \theta_{F_i}, \varphi_{F_i})$ (rad)	$\{\gamma_{F_i} \mid \gamma_{F_i} \in \mathbb{R}, \gamma_{F_i} \leq 2\pi\}$
	$\{\theta_{F_i} \mid \theta_{F_i} \in \mathbb{R}, \theta_{F_i} \leq 2\pi\}$
	$\{\varphi_{F_i} \mid \varphi_{F_i} \in \mathbb{R}, \varphi_{F_i} \leq 2\pi\}$
force magnitude F_{F_i} (N)	$\{F_{F_i} \mid F_{F_i} \in \mathbb{R}, F_{F_i} \leq 10\}$
torque magnitude τ_{F_i} (Nm)	$\{\tau_{F_i} \mid \tau_{F_i} \in \mathbb{R}, \tau_{F_i} \leq 10\}$
6. Gaussian noise on joint velocity signals	On
7. Gaussian noise on F/T sensor signals	On

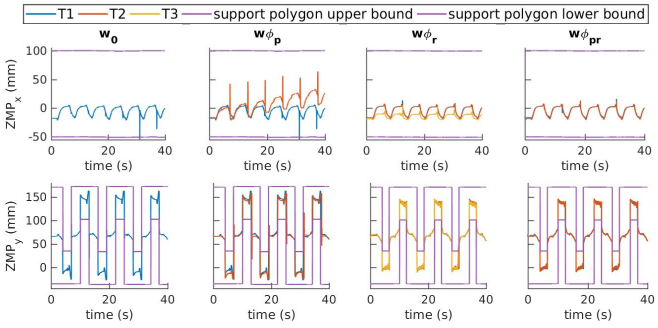


Fig. 4: Typical ZMP trajectories obtained for 6 steps performed with the tethered iCub model, from left to right: given initial weights, weights optimized with ϕ_p, ϕ_r, ϕ_{pr}. Each color (blue, red or yellow) denotes the use of a different set of optimized weights. The x, y and z axes correspond respectively to the sagittal, frontal and vertical axes.

model was subjected randomized conditions 5 to 7 (see Table II), with the testing task to achieve 6 consecutive steps. The success rates achieved with the optimized weights from each fitness function are shown in Table I. Successful CoM and feet trajectories obtained with ϕ_{pr}, using DR and not, are shown in Fig. 3. As could be expected, due to the addition of noise, ZMP trajectories showed to be highly noisy and are therefore not shown. These results show that weights obtained with ϕ_{pr} allow for a higher robustness of the controller. Additionally, the rate of success achieved with DR shows to be significantly higher than without DR, demonstrating that DR did have a measurable impact on the achieved robustness.

V. Discussion and conclusion

In summary, the proposed method generated task priorities for successful whole-body control of different robot models, in 200 learning iterations. It was demonstrated by performing training on a first model of the iCub robot, then testing on a different model submitted to diverse working conditions.

Results achieved with ϕ_p, a fitness function favoring task performance, were likely limited by overfitting on the model used for learning and did not allow much robustness with respect to disturbances. On the other hand, optimizing for robustness with ϕ_r allowed higher chances of success when changing conditions, by encouraging smaller ground reaction forces and the generation of angular momentum through the torso and arms. However, ϕ_r might have neglected the fulfillment of tasks, which are used for keeping balance. Instead, using ϕ_{pr}, a fitness function combining robustness and performance, allowed to achieve superior results.

With ϕ_{pr}, the swing foot placement, crucial for stability at touchdown, was given high importance, while the neck orientation task a lesser one, allowing compliance to external forces, which thus facilitates recovery from external perturbations and contact switching. As for the postural task, its allotted low priority allows it to be used as regularization (just as joint torques), instead of competing with Cartesian tasks. Such a solution is interesting, as it may not have been *a priori* self-evident to an expert defining task priorities. Furthermore, the ranges over which sets of optimized weights were obtained show that the problem has multiple local minima. Therefore, although task priorities require proper tuning, the controller is not highly sensitive to a single precise adjustment of task weights.

Eventually, in order to further improve performance when using a different robot model or moving on to experiments on the real robot, one may be interested in performing another round of optimization on the task priorities, for example exploiting data-efficient learning methods, as done in [11].

Nonetheless, the robustness achieved with the proposed method has the potential to allow higher success when passing from simulation to real-world experiments. Further works shall explore different DR conditions, such as randomizing values associated to the dynamics of the system, control delay, or measurement noise amplitude and delay. The extension of the proposed method to additional parameters such as feedback gains will also be examined.

References

[1] S. Ivaldi, J. Babič, M. Mistry, and R. Murphy, "Special issue on whole-body control of contacts and dynamics for humanoid robots," *Autonomous Robots*, vol. 40, no. 3, pp. 425–428, Mar 2016.

[2] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Souères, 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, April 2013.

[3] G. Nava, F. Romano, F. Nori, and D. Pucci, "Stability analysis and design of momentum-based controllers for humanoid robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 680–687.

[4] S. Dafarra, G. Nava, M. Charbonneau, N. Guedelha, F. Andrade, S. Traversaro, L. Fiorio, F. Romano, F. Nori, G. Metta, and D. Pucci, "A Control Architecture with Online Predictive Planning for Position and Torque Controlled Walking of Humanoid Robots," *ArXiv e-prints*, July 2018.

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

[6] K. Otani, K. Bouyarmane, and S. Ivaldi, "Generating assistive humanoid motions for co-manipulation tasks with a multi-robot quadratic program controller," in *ICRA*, 2018.

[7] N. Dehio, R. F. Reinhart, and J. J. Steil, "Multiple task optimization with a mixture of controllers for motion generation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 6416–6421.

[8] S. Ha and C. Liu, "Evolutionary optimization for parameterized whole-body dynamic motor skills," in *ICRA*, 2016.

[9] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi, "Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization," in *HUMANOIDS*, 2016.

[10] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, pp. 503–507, 2015.

[11] J. Spitz, K. Bouyarmane, S. Ivaldi, and J. B. Mouret, "Trial-and-error learning of repulsors for humanoid qp-based whole-body control," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov 2017, pp. 468–475.

[12] D. Clever, M. Harant, K. D. Mombaur, M. Naveau, O. Stasse, and D. Endres, "Cocomopl: A novel approach for humanoid walking generation combining optimal control, movement primitives and learning and its transfer to the real robot HRP-2," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 977–984, 2017.

[13] S. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 23–30.

[15] R. Antonova, S. Cruciani, C. Smith, and D. Kragic, "Reinforcement learning for pivoting task," *CoRR*, vol. abs/1703.00472, 2017.

[16] J. E. Marsden and T. S. Ratiu, *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Springer Publishing Company, Incorporated, 2010.

[17] R. Olfati-Saber, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, February 2001. [Online]. Available: http://hdl.handle.net/1721.1/8979

[18] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies." *Evolutionary Computation*, vol. 9, pp. 159–195, Jan 2001.

[19] D. V. Arnold and N. Hansen, "A (1+1)-CMA-ES for constrained optimisation," in *GECCO*, 2012, pp. 297–304.

[20] V. Modugno, G. Nava, D. Pucci, F. Nori, G. Oriolo, and S. Ivaldi, "Safe trajectory optimization for whole-body motion of humanoids," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, November 2017, pp. 763–770.

[21] C. Igel, T. Suttorp, and N. Hansen, "A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies," in *GECCO*, 2006.

[22] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The icub humanoid robot: An open platform for research in embodied cognition," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS '08. New York, NY, USA: ACM, 2008, pp. 50–56.

[23] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[24] F. Romano, S. Traversaro, D. Pucci, J. Eljaik, A. Del Prete, and F. Nori, "A whole-body software abstraction layer for control design of free-floating mechanical systems," in *IEEE Int. Conf. on Robotic Computing (IRC)*, 2017, pp. 148–155.