Humanoid Whole-Body Movement Optimization from Retargeted Human Motions

Waldez Gomes[1], Vishnu Radhakrishnan[1], Luigi Penco[1], Valerio Modugno[2],
Jean-Baptiste Mouret[1], Serena Ivaldi[1]

***Abstract*— Motion retargeting and teleoperation are powerful tools to demonstrate complex whole-body movements to humanoid robots: in a sense, they are the equivalent of kinesthetic teaching for manipulators. However, retargeted motions may not be optimal for the robot: because of different kinematics and dynamics, there could be other robot trajectories that perform the same task more efficiently, for example with less power consumption. We propose to use the retargeted trajectories to bootstrap a learning process aimed at optimizing the whole-body trajectories w.r.t. a specified cost function. To ensure that the optimized motions are safe, i.e., they do not violate system constraints, we use constrained optimization algorithms. We compare both global and local optimization approaches, since the optimized robot solution may not be close to the demonstrated one. We evaluate our framework with the humanoid robot iCub on an object lifting scenario, initially demonstrated by a human operator wearing a motion-tracking suit. By optimizing the initial retargeted movements, we can improve robot performance by over 40%.**

I. Introduction

Kinesthetic teaching is widely used to demonstrate the motion required to perform a task to robotics manipulators. Motion retargeting [1], [2] implements this idea for humanoid robots by using human motion tracking data to demonstrate whole-body movements. While this approach is very powerful, it presents two main limitations. First, humans generate movements that may not be optimal for the robot mainly due to structural differences between them: different number of joints, or power generation at the joint level for instance. Second, because of the intrinsic variability of the human motion, the robot may not learn from the best possible representation of the task execution. The main insight is that even if a human demonstration enables the robot to perform a given task, i.e., lift a box (Fig. 1), the retargeted motion will be hardly optimal for the robot from an energetic point of view: if a more efficient way to perform a task exists, it needs to be found.

In this paper, we tackle those issues by using the retargeted motions to bootstrap an optimization process that finds the best whole-body movement that minimizes a given cost function related to the energetic consumption.

Additionally, instead of picking one trajectory, we rely on a parameterized description of the retargeted task trajectories based on Probabilistic Movement Primitives [3], which can capture motion variability computing a Gaussian trajectory distribution from a set of demonstrations. Task trajectories are executed by our multi-task whole-body controller [4] which solves a quadratic programming problem to generate the robot controls for executing the desired motion.

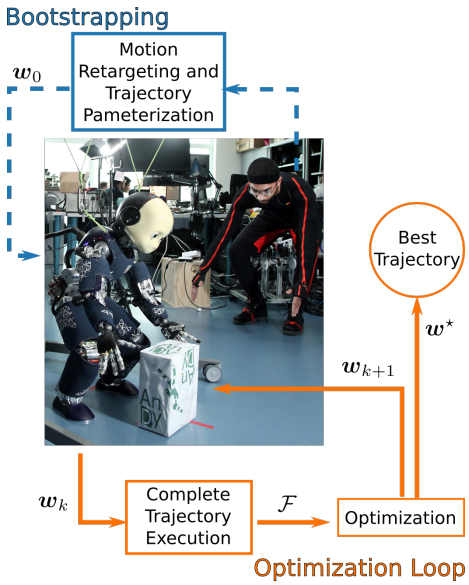

Fig. 1: After several human whole-body demonstrations, the robot learns an initial movement that is further refined by an optimization algorithm in simulation.

As we previously did in [5], we use non-linear constrained optimization to find the best parameters of the whole-body trajectories that minimize an effort cost of executing the desired task, under the constraints related to the robot motion and actuator capabilities. This optimization phase, bootstrapped by the initial retargeted trajectories, evaluates in simulation the execution of the entire trajectory at each episode (roll-out) until an optimal trajectory is found (Fig. 1). We use constrained optimization algorithms to ensure that the solutions are always safe, not violating any constraints [5], [6]. Since we cannot exclude *a priori* that the optimized solution is not close to the initial demonstration, we investigate both local and global constrained optimization algorithms.

We show the effectiveness of our framework for whole-body movement optimization on a lifting task, initially demonstrated by a human operator wearing a motion-tracking suit: the robot performance, evaluated by the fitness related

*This work was supported by the European Unions Horizon 2020 Research and Innovation Programme under Grant Agreement No. 731540 (project AnDy) and from the European Research Council (ERC) under Grant Agreement No. 637972 (project ResiBots).

[1] Inria, CNRS, University of Lorraine, Loria, UMR 7503
[2] La Sapienza University, Rome, Italy

to the torque consumption, improves by over 40%.

The main contribution of this paper is twofold: First, we show a framework that learns optimal whole-body trajectories that do not violate any of the problem constraints, and at the same time benefits from ProMPs to bootstrap the learning with a trajectory that takes into account several human demonstrations. Second, we show the method's viability with different optimization algorithms in a simulated environment, providing practical intelligence for future research on trajectory optimization for humanoid robots.

In the rest of the paper, we explain how similar studies have approached the problem of optimizing motion retargeting for humanoid robots, mainly by optimizing control parameters, or reference trajectories (Section II). Then we follow to the building blocks of our framework (Fig. 2), precisely, fundamentals of our robot controller, ProMPs, our motion retargeting, and the constrained optimization algorithms that were used. In Section IV, we describe our experimental scenario for the humanoid, that it is further evaluated by three different optimization algorithms in a benchmark with different initial trajectories. Finally, we discuss the results in Section V and conclude in favor of the viability of our method to optimize retargeted motions to humanoid robots (Section VI).

II. Related Work

Humanoid robots are able to execute multiple tasks through different joint trajectories (motion redundancy). In order to control a humanoid robot, a well-established approach is to use quadratic programming (QP) formulations that implement strict [7], [8] or soft [9], [10] priorities among many tasks while dealing with low-level constraints (e.g. joint or torque limits). However, even though those QP controllers have shown satisfactory results, they still demand good input trajectories. Otherwise, the high number of degrees of freedom and constraints in humanoid robots may hinder the generation of feasible motions that do not violate robot constraints.

One solution for whole-body trajectory generation is to learn the movements from human demonstrations [11], [12]. In [1], the authors retarget human motion onto a humanoid robot using a QP controller and additionally taking into consideration higher-level constraints such as balance. Other works have achieved motion retargeting in other challenging scenarios, with multiple contacts [2], or heavy object manipulation [13]. Furthermore, demonstrations may be used to learn useful robot control policies. For instance, [14] learn movement constraints from demonstrations, and [15] estimate contact constraints.

The trajectories executed by humans are likely optimal according to one or more criteria related to human motor behavior [16]. However, even after motion retargeting, there is no guarantee that retargeted trajectories are optimal for the robot. In order to execute robot movements optimally, some studies set or define multiple reference trajectories for the robot, and optimize controller parameters. In [17], [18], [19] for instance, the authors parameterize hard or soft priorities for multiple trajectories in a QP controller and posteriorly optimize them w.r.t. to different cost functions related to the humanoid kinematics or dynamics.

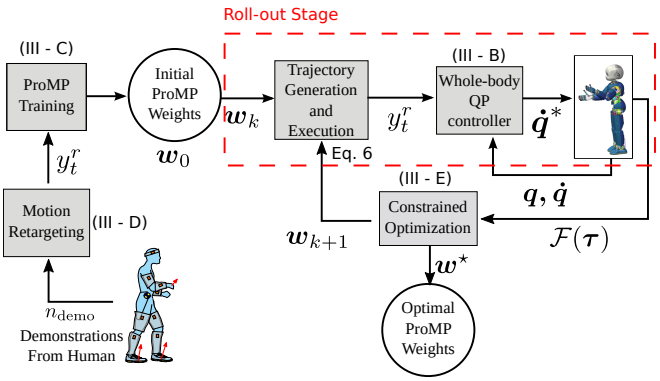

Fig. 2: Overview of our framework for learning optimal weight parameters for whole-body trajectories.

In this work, we take the approach of keeping the QP controller parameters fixed, and optimizing the reference trajectories directly. Furthermore, before trajectory optimization, those trajectories need to be compactly parameterized so that the optimization only deals with a small set of parameters, as it is typically done in motion planning and reinforcement learning [20], [21].

In [22], the authors used reinforcement learning (RL) to modify waypoints in a reference trajectory for the robot's hand. The waypoints were used as parameters for a Bayesian Optimization (BO) framework. It included cost function evaluations directly from robot demonstrations to have only a few roll-outs in comparison to optimizations in simulations.

In [5], the authors parameterized a reference trajectory for the robot Center of Mass (CoM). A compact representation of the entire trajectory is provided through a radial basis function (RBF) network. Furthermore, the trajectory optimization was done in 2 steps: Unconstrained optimization; and black-box constrained optimization, where the solution of the first step was used as an initial point for the second step. The main issue of this study was exactly the fact that its main optimization step required a prior bootstrapping step from a successful unconstrained optimization to guarantee a trajectory that does not violate any constraints. In contrast, here we leverage human demonstrations for bootstrapping the optimization.

In [23], the authors used dynamic movement primitives (DMPs) to parameterize humanoid joint trajectories, and posteriorly optimize them in a RL framework. Similarly, [24] uses DMPs in a hierarchical RL framework that optimizes for sequences of movement, and [25] extended their method to learn the trajectory's end goal and to improve robustness in pick and place applications. The DMPs in all those approaches were trained after task demonstrations by humans.

III. Methods

A. Notation and List of Symbols

Throughout the paper, we use the following notation and list of symbols:

- $n_{bf} \in \mathbb{N}$: Number of basis functions for a ProMP;

- $n_j \in \mathbb{N}$: Number of robot joints;
- $n_{\text{demo}} \in \mathbb{N}$: Number of demonstrations by a human;
- n: Number of tasks being controlled by the QP controller;
- $K \in \mathbb{N}$: Max. number of iterations/roll-outs for the optimization algorithm;
- $T \in \mathbb{N}$: Max. number of time steps during a roll-out;
- $\mathcal{T}_n$: n-th task in a QP problem;
- $\mathcal{S}$: Stack of n tasks $\mathcal{T}$;
- $\boldsymbol{q} \in \mathbb{R}^{n_j+6}$: Position of the robot's joints and floating base pose w.r.t. the origin;
- $y_t \in \mathbb{R}$: Position at instant t for a given one dimensional Cartesian trajectory;
- $\boldsymbol{w}_k \in \mathbb{R}^{n_{bf}}$: Vector of weights for ProMP at roll-out k;
- $\boldsymbol{\mu}_w \in \mathbb{R}^{n_{bf}}$: Mean of weights $\boldsymbol{w}$;
- $\boldsymbol{\Sigma}_w \in \mathbb{R}^{n_{bf} \times n_{bf}}$: Variance of weights $\boldsymbol{w}$;
- $\sigma_y \in \mathbb{R}$: Gaussian noise variance for $y_t(\boldsymbol{w})$
- $\boldsymbol{\phi}_t \in \mathbb{R}^{n_{bf}}$: Basis function vector at instant t;
- $z_t \in [0, 1]$: Phase variable used to decouple a ProMP trajectory from the time signal;
- $\boldsymbol{\tau} \in \mathbb{R}^{n_j}$: Vector of the torques in every joint of the robot;
- $\mathcal{F}(\boldsymbol{\tau}) \in \mathbb{R}$: objective function used in the optimization framework;
- $\boldsymbol{w}^\star$: Weights that optimize $\mathcal{F}$;
- r_i: i-th set of human demonstrations where the human does not bend his/her back;
- s_i: i-th set of human demonstrations where the human bends his/her back;

B. QP Robot Controller and Stack of Tasks

In this work, the execution of the desired tasks of the robot is performed by a velocity-based QP controller [26] (Fig. 2), that enables the execution of multiple tasks at the same time. The control of n tasks $\mathcal{T}$ is formulated as a QP problem:

$$\dot{\boldsymbol{q}}^* = \arg\min_{\dot{q}} \|\boldsymbol{A_n}\dot{\boldsymbol{q}} - \boldsymbol{b_n}\|_{\boldsymbol{W}} \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{C_{1,n}}\dot{\boldsymbol{q}} \leq \boldsymbol{b_{1,n}} \tag{2}$$

$$\boldsymbol{C_{2,n}}\boldsymbol{q} \leq \boldsymbol{b_{2,n}} \tag{3}$$

where $\dot{\boldsymbol{q}}^*$ is the desired velocity sent to low-level controllers (solution for the QP problem), $\boldsymbol{A}_n \in \mathbb{R}^{(n_j+6)\times(n_j+6)}$ is the Jacobian matrix for the task, $\boldsymbol{b}_n \in \mathbb{R}^{(n_j+6)}$ is a reference value for the task, $\boldsymbol{W}$ is a positive definite weight matrix, $\boldsymbol{C}_{c,n} \in \mathbb{R}^{(n_j+6)\times(n_j+6)}$ and $\boldsymbol{b}_{c,n} \in \mathbb{R}^{(n_j+6)}$ encode lower, upper bounds, equalities and inequalities from the c-th constraint for $\dot{\boldsymbol{q}}$.

Conflicts may arise among tasks, so a prioritization scheme is required in the definition of the QP controller. If $\mathcal{T}_1$ has a hard priority over $\mathcal{T}_2$, this means that $\mathcal{T}_2$ is solved in the null space of $\mathcal{T}_1$, and the stack is represented as: $\mathcal{S} = \mathcal{T}_1/\mathcal{T}_2$.

If two tasks have a soft priority relationship, we can represent this as a new task: $\mathcal{T}_3 = \mathcal{T}_1 + \mathcal{T}_2$, where the new task $\mathcal{T}_3$ is a weighted norm of $\mathcal{T}_1$ and $\mathcal{T}_2$.

Using the notation above, our robot is controlled according to the stack $\mathcal{S}_1$:

$$\begin{aligned}\mathcal{S}_1 = (\mathcal{T}_{LeftFoot} + \mathcal{T}_{RightFoot} + \mathcal{T}_{head} + \mathcal{T}_{waist}) \,/ \\ (\mathcal{T}_{LeftHand} + \mathcal{T}_{RightHand} + \mathcal{T}_{CoM})\end{aligned} \tag{4}$$

where the tasks above are responsible for the Cartesian trajectories of the left foot, right foot, waist, left hand, right hand, and center of mass of the robot. Only the head task is a joint trajectory task. For more details on the QP controller implemented by the OpenSoT library please refer to [27].

C. Probabilistic Movement Primitives (ProMPs)

Movement Primitives (MPs) are often used to represent parameterized trajectories, learned via demonstrations. Probabilistic Movement Primitives (ProMPs) have a probabilistic formulation that catches the variance of the demonstrations. Additionally, among other properties, it is possible to modulate a ProMP both in time and space without deforming its shape [28].

A ProMP can compactly represent trajectories with a simple weight vector. For instance, a one-dimensional trajectory $y_t \in \mathbb{R}$ can be represented with a weight vector $\boldsymbol{w}$:

$$y_t(\boldsymbol{w}) = \boldsymbol{\phi}_t^\top \boldsymbol{w} + \epsilon_y \tag{5}$$

and the probability of observing this trajectory given a certain vector $\boldsymbol{w}$ is given as a linear basis function model:

$$p(y_t|\boldsymbol{w}) = \prod_t \mathcal{N}(y_t|\boldsymbol{\phi}_t^\top \boldsymbol{w}, \sigma_y) \tag{6}$$

where the variable $\epsilon_y \sim \mathcal{N}(0, \sigma_y)$ is a zero-mean i.i.d Gaussian noise, and $\boldsymbol{\phi}_t \in \mathbb{R}^{n_{bf}}$ is a time dependent basis function vector for the trajectory positions. The general form of the i-th basis function b_i is given by:

$$b_i = \exp\left(-\frac{(z_t - c_i)^2}{2h}\right) \tag{7}$$

where h defines the basis' width and c_i is the basis' center. The basis' centers are uniformly distributed in $[-2h, (1 + 2h)]$. Then, we normalize the basis functions

$$\phi_t^i = \frac{b_i}{\sum_{j=1}^n b_j} \tag{8}$$

to maintain the same summed activation. In other words, y_t is represented as a weighted sum of n_{bf} normalized basis functions uniformly distributed in time.

To capture the variability of the movement, $\boldsymbol{w}$ is also represented as a Gaussian variable, $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w})$. Therefore, by marginalizing out the weight vector in 6 we get:

$$\begin{aligned}p(y_t|\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}) &= \int \mathcal{N}(y_t|\boldsymbol{\phi}_t^\top \boldsymbol{w}, \sigma_y)\mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w})d\boldsymbol{w} \\ &= \mathcal{N}(y_t|\boldsymbol{\phi}_t^\top \boldsymbol{\mu_w}, \boldsymbol{\phi}_t^\top \boldsymbol{\Sigma_w}\boldsymbol{\phi}_t + \sigma_y)\end{aligned} \tag{9}$$

Before using Eq. (9), we need to train the ProMP, that is, we need to estimate a mean ($\boldsymbol{\mu_w}$) and variance ($\boldsymbol{\Sigma_w}$) for the variable $\boldsymbol{w}$ based on n_{demo} trajectory observations. Similarly

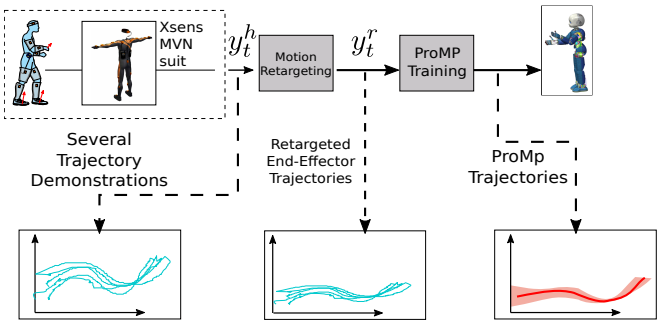

Fig. 3: Process of extracting trajectories from demonstrations: 1) Acquisition of Cartesian position trajectories from a human wearing an Xsens suit; 2) Scaling the position trajectories for the iCub robot; 3) Training a ProMP, smoothing the raw data and associating it with a variance of motion.

to [28], we use a maximum likelihood estimation algorithm alongside a linear ridge regression to achieve the training.

By using ProMPs as trajectory references instead of the demonstrations directly, we are able to smooth the trajectories as well as capture their variability (Fig. 3).

D. Motion Retargeting

The human that executed the demonstrations and the iCub robot have different kinematic structures. Therefore, human trajectories have to be retargeted into feasible corresponding values for the robot. This is achieved by either retargeting the joint trajectories or by carefully choosing links and retargeting their Cartesian positions. Here, we chose the latter to decrease the number of trajectories that need to be retargeted to represent a whole-body task.

A way to retarget one link position is to consider its relative position w.r.t. a base link, and measure the length of such limb in both the human and the robot [1]. The relative position of the robot end effector is then computed as:

$$y_t^r = k_{rtg}\left(y_t^h - y_0^h\right) + y_0^r \tag{10}$$

where $k_{rtg} \in \mathbb{R}$ is the ratio between the robot and the human limb lengths, r and h are superscripts related to the robot or human respectively, and 0 is a subscript indicating that this is the position for an initial instant.

Effectively, Eq. (10) scales down the trajectory from the human to the child sized iCub robot (Fig. 3).

E. Constrained Optimization

In our framework (Fig. 2) we optimize the parameters of many ProMP trajectories $y_t(\boldsymbol{w})$ w.r.t. an objective function $\mathcal{F}(\boldsymbol{\tau})$, that is a function of the torques for each trajectory execution. In other words, we want to find a set of weights $\boldsymbol{w}^\star$ that generates a set of trajectories $\boldsymbol{y}_t^\star$ that minimizes $\mathcal{F}$.

For every set of weights, $\boldsymbol{w}_k$, a new cost function is computed after the robot simulation in a physics engine, and an optimization algorithm selects a new set of weights $\boldsymbol{w}_{k+1}$ to explore the space of $\boldsymbol{w}$. This approach is equivalent to an episodic learning loop with the maximum number of K roll-outs (or episodes).

One of the contributions of this work is to use the ProMP trained from human demonstrations to bootstrap the search for an optimal trajectory. It is assumed that the demonstrations from humans for a given activity are already close to minimize Eq. (13). Additionally, to increase the positive influence on the search for $\boldsymbol{w}^\star$, we add sets of high-level non-linear constraints such as Robot must be standing up at the end of trajectory execution; and the robot must reach a set of fixed targets in space during the trajectory. We also add bound constraints to $\boldsymbol{w}$ so that the trajectories do not reside outside of the robot workspace (further explained in the Experiments section).

Due to the derivative-free nature of the problem, non-linear constraints optimization algorithms are used to solve it. In the next section, we briefly describe the different optimization algorithms used on our benchmark for a given task.

F. Optimization Algorithms

Three optimization algorithms were used to evaluate our proposed method:

1) COBYLA: (Constrained Optimization BY Linear Approximation) is a deterministic algorithm that constructs linear approximations of the cost function and then optimizes them at each roll-out [29].

2) AGS: is a deterministic algorithm proven to converge to a global optimum if the cost function satisfies the Lipschitz condition within the bound constraints [30].

3) CCMA-ES: (short for (1+1)-CMA-ES with Constrained Covariance Adaptation) is a stochastic optimization algorithm. At every roll-out, it evaluates a set of samples drawn from a multivariate Gaussian distribution. If the samples violate any of the previously set constraints, it adapts the covariance matrix of the distribution and re-samples from the new co-variance. This design ensures that the constraints are never violated [31]. Furthermore, its design also necessarily demands that it has to start with a point that does not violate any constraints, otherwise the algorithm will get stuck and fail. This algorithm has already been used in an earlier work for learning soft task priorities for the iCub robot [17].

We used the NLopt library [32] implementations for COBYLA and AGS and our C++ version of CCMA-ES.

IV. Experiments

A. Task Description and Optimization Parameters

We defined a pick and place experiment where a human/iCub has to squat, grab a box, and stand up (Fig. 4). As explained in the above section, we recorded a human executing the target trajectory to define a starting solution for the robot. In real life, the human grabbed the box 9 times with variants of movement strategies. In particular, for the first 4 movements, the subject did not bend its back, while for the other 5 it does bend its back.

To control the robot with the stack $\mathcal{S}_1$, we need the trajectories for the left hand, right hand, waist, and center of mass. The other tasks, head, left foot, and right foot are fixed. The controlled trajectories are:

- ${}^{Z}\boldsymbol{y}_t^{Waist}$, Z-Axis of the waist;
- ${}^{X}\boldsymbol{y}_t^{CoM}$, X-Axis of the CoM;

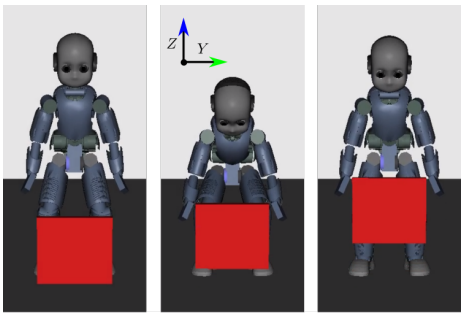

Fig. 4: In our designed task, iCub has to squat, grab a box, and then stand up with the box.

- ${}^{X}\boldsymbol{y}_t^{RH}, {}^{Z}\boldsymbol{y}_t^{RH}$, X- and Z-Axis of the right hand;
- ${}^{X}\boldsymbol{y}_t^{LH}, {}^{Z}\boldsymbol{y}_t^{LH}$, X- and Z-Axis of the left hand;

however, to simplify the optimization using less parameters we assume the right hand trajectories to be symmetrical to the left hand

$$ {}^{X}\boldsymbol{y}_t^{RH} = {}^{X}\boldsymbol{y}_t^{LH} \tag{11} $$
$$ {}^{Z}\boldsymbol{y}_t^{RH} = {}^{Z}\boldsymbol{y}_t^{LH} \tag{12} $$

In this manner, we only need to learn a ProMP for the right hand, and mirror it like in Eq. (12), totaling 4 ProMPs to be learned in order to execute $\mathcal{S}_1$.

Our framework is independent of the choice of a cost function $\mathcal{F}$, however, the choice of $\mathcal{F}$ is how we define what is a good movement. Here, we define $\mathcal{F}$ such as to minimize efforts at the hips, torso, and shoulders:

$$ \mathcal{F}(\boldsymbol{\tau}) = \sum_{t}^{T} \frac{1}{n_j} \sum_{i=1}^{n_j} \left(\frac{v_i \tau_i}{\tau_i^{\max}} \right)^2 \tag{13} $$

where T is the number of time steps for each roll-out, $\tau_i^{\max}$ is a known maximum value for the torque at the i-th joint, and v_i is a fixed weight for the i-th joint. Eq. (13) defines a sum of squared torques at every joint, therefore, as $\mathcal{F}$ decreases, so does the effort applied by the robot actuators. The weights are defined $v_i = 1$ for every joint, except for the high priority and low priority joints. The high priority joints with $v_i = 5$ are: left hip, right hip, torso, left shoulder, right shoulder. Furthermore, the low priority joints with $v_i = 0.1$ are: Left knee, right knee, left ankle, and right ankle.

As for the optimization parameters, we defined $n_{bf} = 5$, so for each ProMP there are 5 weight variables to be optimized. However, in order to further decrease the number of optimization parameters, we decided to optimize only for the CoM and waist trajectories (The Cartesian retargeted motions are still learned as ProMPs). It is important to note that the reduction of optimization parameters is taken solely to speed up the optimization computational process. Finally, the optimization variables are

$$ \boldsymbol{w} = \left[(\boldsymbol{w}^{waistZ})^\top, (\boldsymbol{w}^{comX})^\top \right]^\top \in \mathbb{R}^{10} \tag{14} $$

TABLE I: Description of initial sets used on the benchmark.

Initial Set ID	Description
$r_1 - r_4$	Single Demonstration(s) w/ straight torso
r_{all}	ProMP from r_1, r_2, r_3, r_4
$s_1 - s_5$	Single Demonstration(s) w/ bent torso
s_{all}	ProMP from s_1, s_2, s_3, s_4, s_5
all	ProMP from every demonstration

where $\boldsymbol{w}^{waistZ}$ is bound between 0 and 1 and $\boldsymbol{w}^{comX}$ is bound between -0.5 and 0.5, effectively, not allowing the Z component of the CoM trajectory to go below the inertial reference (ground) and the X component of the waist to go beyond 0.5 meters of the inertial reference.

Besides the standing-up at all times constraint, iCub's hands have to achieve a specific set of targets at a particular order for the task to be considered executed. First, they have to reach the grasping box position):

$$ {}^{X}\boldsymbol{y}_t^{RH}, {}^{X}\boldsymbol{y}_t^{LH} > 0.15 \tag{15} $$
$$ {}^{Z}\boldsymbol{y}_t^{RH}, {}^{Z}\boldsymbol{y}_t^{LH} < 0.30 \tag{16} $$

and then they have to return to an initial position:

$$ {}^{Z}\boldsymbol{y}_t^{RH}, {}^{Z}\boldsymbol{y}_t^{LH} > 0.40 \tag{17} $$

If the robot's hands do not reach those goals, then the entire execution is said unsuccessful, and a black-box constraint is violated. Furthermore, inside the simulation environment, whenever Eq. (15), (16) are satisfied for the first time, a virtual box of 1 Kg is added to the robot, emulating a grasp. Note that the above inequalities are constant. This is done in order to fix the same kinematic task for different ProMP sets, allowing us to compare their performance.

B. Benchmark for Learning $\boldsymbol{w}^\star$ in Simulation

To demonstrate that our framework (Fig. 2) does not require a very specific ProMP as a starting point, different sets of demonstrations are used at the ProMP learning stage (Table I). Where the r_i sets correspond to demonstrations where the human did not bend his/her back to pick up the box, while the s_i sets correspond to the ones where the human did bend his/her back.

For AGS, the number of roll-outs was set to $K = 2000$. On the other hand, CCMA-ES and COBYLA would always converge to a solution much earlier (Fig. 5). Therefore, the number of roll-outs for them was reduced to $K = 500$. The hyperparameters for the CCMA-ES algorithm were set to $\sigma = 0.1$, $\lambda = 1$ (default values from the benchmarks in the original paper [31]). The hyperparameters for AGS and COBYLA were default values in the NLOPT library [32].

To evaluate the performance of the algorithms in our optimization framework we measured the cost function value at the starting point (straight from the motion retargeting) and after a $w^\star$ was found. This gives rise to 2 measurements: the cost function $\mathcal{F}$, and the improvement function

$$ \mathcal{I} = 1 - \frac{\mathcal{F}^\star}{\mathcal{F}_{initial}} \tag{18} $$

TABLE II: Final $\mathcal{F}$ scores after optimization, and $\mathcal{I}$ compared with the initial score for each optimization algorithm. The initial sets s_1-s_{all} are not able to complete the task of picking up the box, therefore, it is not possible to compute an initial score for the initial $\mathcal{F}$ and for the improvement $\mathcal{I}$ (marked with a /). Furthermore, CCMA-ES will also fail for them because the initial points violate the task constraints (marked with a //).

		AGS, $K = 2000$		COBYLA, $K = 500$		CCMA-ES, 30 iterations with $K = 500$	
Initial Set	**Initial $\mathcal{F}$**	$\mathcal{F}^\star$	$\mathcal{I}$	$\mathcal{F}^\star$	$\mathcal{I}$	$\mathcal{F}^\star$ **Median and IQR**	$\mathcal{I}$ **Median and IQR**
r_1	188.55	188.55	00.00%	121.32	35.65%	116.12 (110.55 - 125.84)	38.41 (33.26 - 41.37)
r_2	229.72	215.21	6.32 %	125.01	45.58%	134.16 (127.97 - 143.32)	41.60 (37.61 - 44.29)
r_3	249.63	198.89	20.32%	202.90	18.71%	116.10 (105.10 - 145.40)	53.49 (41.76 - 57.90)
r_4	209.60	209.60	00.00 %	116.80	44.27 %	120.41 (109.50 - 134.24)	42.55 (35.96 - 47.76)
r_{all}	203.11	203.11	00.00%	102.68	49.44 %	114.94 (97.93 - 128.18)	43.41 (35.89 - 51.78)
s_1	/	213.15	/	412.61	/	//	/
s_2	/	202.47	/	350.33	/	//	/
s_3	/	213.68	/	409.34	/	//	/
s_4	/	213.55	/	279.12	/	//	/
s_5	/	186.55	/	206.68	/	//	/
s_{all}	/	223.28	/	383.62	/	//	/
all	223.52	223.52	00.00 %	120.71	45.99 %	111.80 (96.79 - 128.21)	49.98 (42.64 - 56.70)

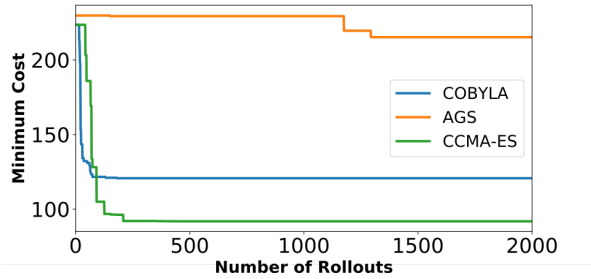

Fig. 5: In our problem, COBYLA and CCMA-ES converge to a solution much faster than AGS. The figure shows the optimization of set r_2

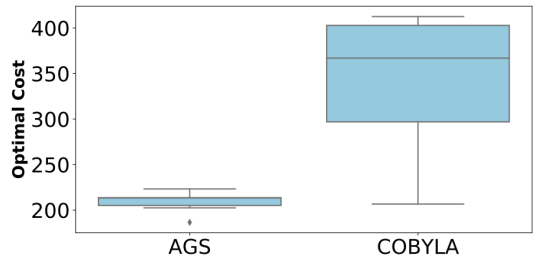

Fig. 6: Optimal cost function from unfeasible starting sets $(s_1, s_2, s_3, s_4, s_5, s_{all})$.

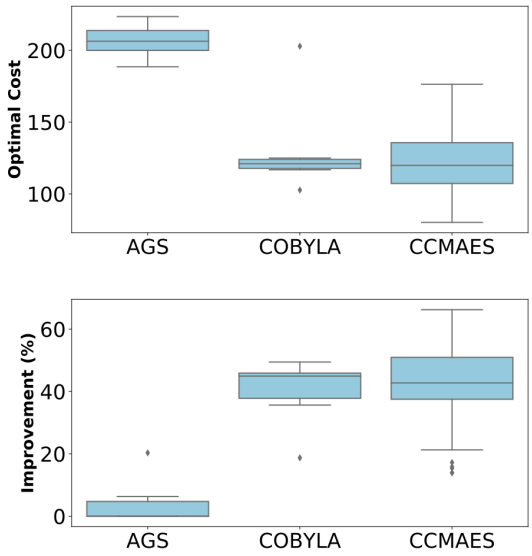

Fig. 7: Optimal cost function, and improvement from feasible starting sets $(r_1, r_2, r_3, r_4, r_{all}, all)$.

C. Results

The results for all optimization algorithms are displayed in Table II. AGS and COBYLA are deterministic algorithms, so they were executed only once for every initial set. Whereas CCMA-ES is a stochastic algorithm, thus, in order to better evaluate the performance of the algorithm, it was executed 30 times. Additionally, we also show the torque values at the high priority joints (hips, torso, and shoulder) (Fig. 9).

During the execution of the benchmark, we found out that the sets $s_1 - s_{all}$ did not pick up the box successfully according to Equations (15),(16),(17). Therefore, the initial cost function and improvements for those measures are inexistent. Moreover, those sets violated the task constraints from the start, so CCMA-ES cannot find a solution for them. However, contrarily to CCMA-ES, both AGS and COBYLA were able to find optimal values even when starting with unfeasible starting points. This different behavior within the sets of initial trajectories demands for a more careful comparison between the algorithms. For this reason, the analysis is first done concerning the unfeasible starting trajectories (Fig. 6) and posteriorly concerning the feasible starting trajectories (Fig. 7).

In Fig. 8, we show the optimized reference trajectories sent to the controller, for the different algorithms, compared with the initial solution with the ProMP that combined every demonstration, all. We do not show the result for AGS, as it behaved poorly when compared to the other ones (Table II).

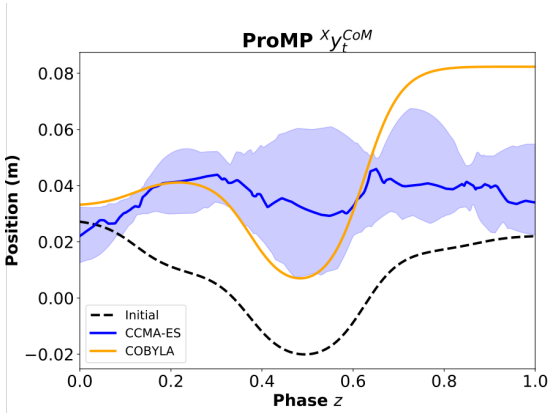

Fig. 8: Reference trajectories: initial (from human demonstrations) and after optimization. Both CCMA-ES and COBYLA were bootstrapped from a ProMP that combined every demonstration, all. The 30 reference trajectories for CCMA-ES are represented with their median and IQR.

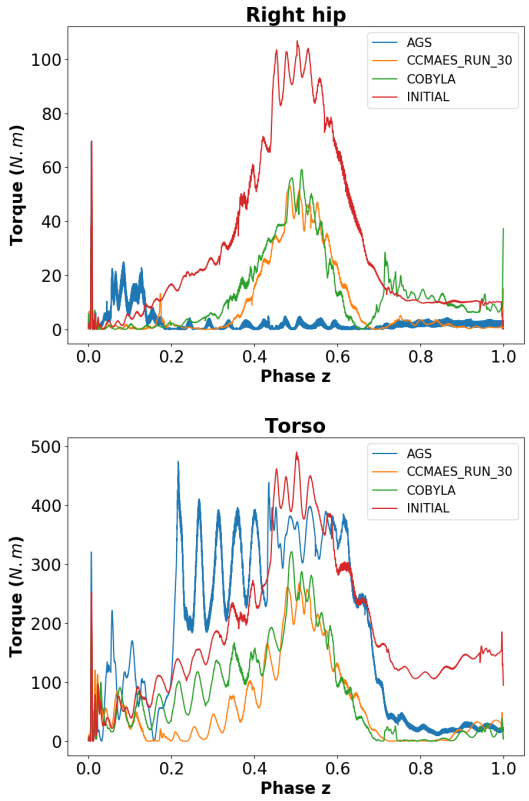

Fig. 9: Torque values at the most critical joints in $\mathcal{F}$.

V. Discussion

The AGS algorithm was only able to improve 2 feasible initial sets, with very little improvement (Table II). However, AGS was remarkably able to find feasible solutions when starting from unfeasible points. These can be explained by AGS thoroughly and rapidly exploring the space, therefore, it is able to find a solution but it is not able to refine it.

COBYLA provided low-cost functions when starting from feasible solutions (Table II), and like AGS, it also managed to find solutions even when starting from trajectories that violated constraints. It is noticeable, though, that COBYLA performs worse than AGS in the latter condition (Fig. 6). This is likely caused by COBYLA being a local optimization algorithm, that works well at refining good solutions, but it does not perform well at exploring the search space.

Over 30 iterations, CCMA-ES was able to locate very good solutions in some of the initial sets (noticeably r_{all}, and all). Additionally, all results from within the Interquartile Range (IQR) are better than the initial starting sets (Table II). Interestingly, some of the solutions from CCMA-ES output a behavior that is known to be non-ergonomic for humans, e.g., bending the back while lifting, as shown in the video attachment. While this behavior is not desirable for humans, it is coherent with the fitness for which it is optimized on the robot. This is further indication that our framework indeed is a step towards efficient motion optimization after retargeting from humans to humanoid robots.

When comparing the results from all three algorithms (Fig. 7) it is possible to verify that COBYLA and CCMA-ES have a very similar median behavior with improvements around the range of 40%. However, often, only the best trajectory is needed, and for this matter, CCMA-ES is more suitable, as their best results are better than the ones from COBYLA. AGS did not seem to produce good results even for 4 times more roll-outs than the other algorithms. Lastly, we can also verify that the cost function $\mathcal{F}$ was able to minimize the torques at all of the high priority joints (Fig. 9).

In this work, we used a cost function that prioritized minimizing torques at selected joints. However, it is possible to define different weights (v_i) at Eq. (13) and obtain different kinds of movements. Additionally, other cost functions like in Charbonneau et al. [6] can favor different movement aspects.

VI. Conclusions

Retargeting the motion captured from a human onto a humanoid robot provides solutions that may not be efficient for the robot. Our framework optimized retargeted trajectories of whole-body motions with constrained optimization algorithms. And further tested this framework in a benchmark with three distinct black-box optimizers: AGS (deterministic and

global), COBYLA (deterministic), and CCMA-ES (Stochastic). Among the tested algorithms, CCMA-ES seems to be the best option, as it combines local and global exploration, even though it also requires starting points that do not violate any constraints, as we had already observed in [5].

Our approach can be extended with different cost functions, other than effort, to harness different movement behaviors from the robot. In addition, a possible future application of this work could include optimizing the design of a robot to accomplish tasks that were initially demonstrated by humans. This is possible mainly due to using Cartesian reference trajectories and representing them using mean trajectories of ProMPs in a compact form so that they can be optimized.

Additionally, the ProMP representation of trajectories is able to do more than representing a mean trajectory and modulating its duration without deforming its shape. For instance, it is possible to combine ProMPs to tackle more than one task at the same time. In order to approach these scenarios, future work will have to also include learning variance of motion, and not only learning the trajectory.

Our framework is inherently based on physical simulations, and therefore, it may fail on tests with real robots, especially if the simulation environment and the real world have large discrepancies. A possible solution would be to input our optimized trajectories as priors for a reinforcement learning approach that deals with the real robot and its environment.

REFERENCES

[1] L. Penco, B. Clement, V. Modugno, E. Mingo Hoffman, G. Nava, D. Pucci, N. G. Tsagarakis, J. . Mouret, and S. Ivaldi, "Robust real-time whole-body motion retargeting from human to humanoid," in 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids).

[2] A. Di Fava, K. Bouyarmane, K. Chappellet, E. Ruffaldi, and A. Kheddar, "Multi-contact motion retargeting from human to humanoid robot," in 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids).

[3] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in Advances in Neural Information Processing Systems 26, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013.

[4] E. M. Hoffman, B. Clement, C. Zhou, N. G. Tsagarakis, J.-B. Mouret, and S. Ivaldi, "Whole-Body Compliant Control of iCub: first results with OpenSoT," in ICRA 2018 - Workshop.

[5] 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).

[6] M. Charbonneau, V. Modugno, F. Nori, G. Oriolo, D. Pucci, and S. Ivaldi, "Learning robust task priorities of qp-based whole-body torque-controllers," in 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids).

[7] A. D. Prete, F. Nori, G. Metta, and L. Natale, "Prioritized motion-force control of constrained fully-actuated robots: task space inverse dynamics," RAS, vol. 63, pp. 150 – 157, 2015.

[8] A. Escande, N. Mansard, and P. B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in 2010 IEEE International Conference on Robotics and Automation.

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

[10] 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, pp. 1283–1290.

[11] S. Schaal, "Is imitation learning the route to humanoid robots?" Trends in cognitive sciences, vol. 3, no. 6, pp. 233–242, 1999.

[12] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in Third IEEE International Conference on Humanoid Robotics 2003, Germany, 2003.

[13] K. Otani and K. Bouyarmane, "Adaptive whole-body manipulation in human-to-humanoid multi-contact motion retargeting," in 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids).

[14] H. Lin, M. Howard, and S. Vijayakumar, "Learning null space projections," in 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2613–2619.

[15] V. Ortenzi, H. Lin, M. Azad, R. Stolkin, J. A. Kuo, and M. Mistry, "Kinematics-based estimation of contact constraints using only proprioception," in 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids).

[16] S. Ivaldi, O. Sigaud, B. Berret, and F. Nori, "From humans to humanoids: The optimal control framework," Paladyn, vol. 3, no. 2, pp. 75–91, Jun 2012.

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

[18] K. Yuan, I. Chatzinikolaidis, and Z. Li, "Bayesian optimization for whole-body control of high-degree-of-freedom robots through reduction of dimensionality," IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 2268–2275, July 2019.

[19] J. Silverio, S. Calinon, L. Rozo, and D. G. Caldwell, "Learning task priorities from demonstrations," IEEE Transactions on Robotics, vol. 35, 2019.

[20] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in 2010 IEEE/RSJ international conference on intelligent robots and systems. IEEE.

[21] B. Price and C. Boutilier, "Accelerating reinforcement learning through implicit imitation," Journal of Artificial Intelligence Research, vol. 19, 2003.

[22] R. Lober, V. Padois, and O. Sigaud, "Efficient reinforcement learning for humanoid whole-body control," in 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids).

[23] F. Stulp, J. Buchli, E. Theodorou, and S. Schaal, "Reinforcement learning of full-body humanoid motor skills," in 2010 10th IEEE-RAS International Conference on Humanoid Robots.

[24] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in 2011 11th IEEE-RAS International Conference on Humanoid Robots.

[25] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," IEEE Transactions on Robotics, vol. 28, no. 6, 2012.

[26] E. Mingo Hoffman, A. Rocchi, A. Laurenzi, and N. G. Tsagarakis, "Robot control for dummies: Insights and examples using opensot," in 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), Nov 2017, pp. 736–741.

[27] A. Rocchi, E. Mingo Hoffman, D. G. Caldwell, and N. G. Tsagarakis, "Opensot: A whole-body control library for the compliant humanoid robot coman," in 2015 IEEE International Conference on Robotics and Automation (ICRA).

[28] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," Autonomous Robots, vol. 42, no. 3, pp. 529–551, Mar 2018.

[29] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in Advances in optimization and numerical analysis. Springer, 1994, pp. 51–67.

[30] Y. D. Sergeyev and D. L. Markin, "An algorithm for solving global optimization problems with nonlinear constraints," Journal of Global Optimization, vol. 7, no. 4, pp. 407–419, 1995.

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

[32] S. G. Johnson, The NLopt nonlinear-optimization package. [Online]. Available: http://ab-initio.mit.edu/nlopt