Learning task controllers on a humanoid robot using multi-objective optimization

Evelyn d'Elia, Jean-Baptiste Mouret, Jens Kober, Serena Ivaldi

Learning task controllers on a humanoid robot using multi-objective optimization

Evelyn D'Elia[1,2], Jean-Baptiste Mouret[1], Jens Kober[2], Serena Ivaldi[1]

***Abstract*— Designing controllers for complex robots is not an easy task. Often, researchers hand-tune controllers for humanoid robots, but this approach requires lots of time for a single controller which cannot generalize accurately to varied tasks. We present a method which uses a multi-objective evolutionary algorithm with various training trajectories and outputs a diverse set of well-functioning controller weights and gains. The results of this optimization in the Talos robot show which weight and gain ranges can be used for a robust controller and prove that the optimization yields a diverse set of controller parameters, many of which can succeed on modified robot models.**

I. INTRODUCTION

It is difficult to find controllers that enable a humanoid robot to perform a variety of different loco-manipulation trajectories. This problem can be simplified by breaking it down into a set of tasks, each with certain gain and weight parameters to define their behavior. This is called a task priority-based approach, as explained in [1], and it makes controller design more straightforward, but it still requires the task parameters to be chosen. One way to choose them is by hand-tuning these parameters, but since this method is time-consuming, it often focuses on tuning a single controller which works for a large set of trajectories. The drawback of this method is that with a single "robust" controller, the quality of individual trajectories is compromised.

To save time, and to ensure that there are highly accurate controllers for specific types of trajectories, we propose the use of NSGA-II, a multi-objective optimization algorithm, to learn a Pareto front of task parameter sets for the Talos robot (shown in 1) that are scored based on how they perform for each of a set of training trajectories. This method is similar to that used in [2], whose approach is based only on average accuracy and stability over the training trajectories which produces a less diverse Pareto front. Instead, our approach directly optimizes based on each trajectory performance.

II. METHODS

The control framework on which the Talos runs is based on quadratic programming (QP) with soft task priorities. At each time step, the following optimization problem is solved:

$$\begin{aligned} \min_{u} \quad & \sum_{n} w_n \left(\|\boldsymbol{A}_n \boldsymbol{u} - \boldsymbol{b}_n\|^2 + \epsilon \|\boldsymbol{u}\|^2 \right) \\ \text{s.t.} \quad & \boldsymbol{c}_{min} \leq \boldsymbol{C}\boldsymbol{u} \leq \boldsymbol{c}_{max} \\ & \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max}, \end{aligned} \tag{1}$$

where $\boldsymbol{A}_n$ is the task's equivalent Jacobian, $\boldsymbol{u}$ is the control input in the form of joint torques, and ϵ is a regularization factor, used to avoid singularities. The constraints bound the control input, dynamics, and environmental factors. We use inverse dynamics (ID) control, therefore the task reference $\boldsymbol{b}_n$ is defined as:

$$\boldsymbol{b}_n = \ddot{\boldsymbol{p}}_n^d - \dot{\boldsymbol{J}}_n \dot{\boldsymbol{q}} + \lambda_n^P \boldsymbol{e} + \lambda_n^D \dot{\boldsymbol{e}}, \tag{2}$$

where n is the task index, $\ddot{\boldsymbol{p}}_n^d$ is the desired task acceleration, $\dot{\boldsymbol{J}}_n$ is the derivative of the task Jacobian, $\dot{\boldsymbol{q}}$ are the joint velocities, and λ_n^P and λ_n^D are proportional and derivative scalar convergence gains (CG) for the task position $\boldsymbol{e}$ and velocity $\dot{\boldsymbol{e}}$ errors respectively. Soft priorities are represented by a soft priority weight (SPW) w_n on the error of each task $\mathcal{T}_n$. The tasks we consider are shown in Table I. We use 5 distinct tasks, since those of the hands and feet are the same on both sides. In total there are 10 parameters that are optimized, represented by the parameter vector $\boldsymbol{\theta}$.

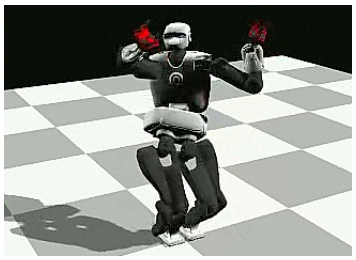

Fig. 1. Simulated Talos performing a dance, recorded from human movements using the Xsens MVN motion tracking suit.

TABLE I
SYMBOL, DESCRIPTION, SPW NAME AND CG TYPE FOR EACH TASK CONSIDERED.

Task	Description	SPW	CG
$\mathcal{T}_{\{rh,lh\}}$	hand pose (symmetric)	w_h	$\lambda_h^P = \sigma_h^P$
$\mathcal{T}_{\{rf,lf\}}$	foot pose (symmetric)	w_f	$\lambda_f^P = \sigma_f^P$
$\mathcal{T}_{CoM}$	CoM position	w_{CoM}	λ_{CoM}^P
$\mathcal{T}_{to}$	torso orientation (roll, pitch)	w_{to}	σ_{to}^P
$\mathcal{T}_p$	joint angles	w_p	μ_p^P

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 731540 (project AnDy) and partly funded by the CPER project "CyberEntreprises" and the CPER project SCIARAT.

[1]Inria, University of Lorraine, CNRS, Loria, Nancy, France evelyn.d-elia@inria.fr

[2]Delft University of Technology, Delft, Netherlands

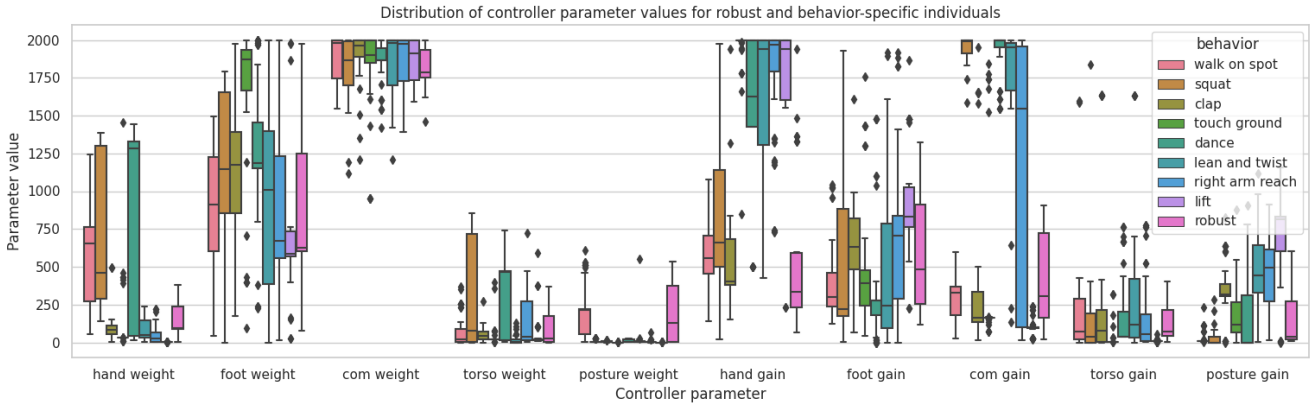

Fig. 2. Comparison of learned control parameters for 8 training trajectories and for robust controllers (those that work on all trajectories). Specific trajectory results based on best 50 controllers among 5 different datasets, and robust results based on all robust controllers in 5 datasets (26 total).

TABLE II

AVERAGE CARTESIAN TASK ERROR ACHIEVED FOR EACH TRAINING TRAJECTORY BY A HAND-TUNED CONTROLLER VERSUS THE AVERAGE OBJECTIVE SCORE OF THE ROBUST CONTROLLERS LEARNED WITH OUR METHOD.

Trajectory	Hand-tuned (m)	Learned (m)
Walk on spot	0.0923	0.1020
Squat	0.1196	0.0894
Clap	0.4375	0.4105
Touch ground	0.4433	0.4242
Dance	0.1970	0.1898
Lean and twist	0.1889	0.1597
Right arm reach	0.1881	0.1865
Lift	0.1881	0.1770

We use NSGA-II [3], one of the best-performing multi-objective evolutionary optimization algorithms, to generate a set of Pareto optimal parameter sets, for which no one objective function can be further optimized without compromising the performance of another objective [4]. In this work we use 8 objective functions, where one objective is calculated per trajectory. Each objective function is given as:

$$f_b = \frac{1}{N_t} \sum_{t=0}^{T} \left(\left\| \boldsymbol{e}_t^{CoM} \right\| + \left\| \boldsymbol{e}_t^{hands} \right\| + \left\| \boldsymbol{e}_t^{feet} \right\| \right), \quad (3)$$

where b is the training trajectory number, N_t is the total number of time steps in the trajectory, T is the end time, and the 3-D Cartesian position error $\boldsymbol{e}$ is calculated for the center of mass (CoM), the hands, and the feet.

To determine whether this method yields transferable controller solutions, we created 5 slightly different robot models by varying the mass: an extra 5 kg in the right hand, left shoulder, and as a backpack on the robot, and overall weight scaled up and down 10% respectively.

III. RESULTS

We evaluate each set of task parameters using its performance in Equation 3. We can see from Table II that the average "robust" controller learned via our method achieves a lower Cartesian tracking error than the robust controller tuned by a human expert. A video of the learned trajectories is available at: https://youtu.be/cnIo-aWCOcs.

For each of the modified robot models we tested, we found that there were at least a few successful controllers for each trajectory, which shows the Pareto front is diverse enough to succeed when the actual robot used is different from the one which the Pareto front was designed for (which is inevitable for the physical robot).

From Figure 2, we see from the universally high CoM weights and low posture weights that no matter the trajectory, the CoM and posture tasks are always the most and least important, respectively. Some of the individual trajectory results are surprising, such as that the hand weight is high for the squat trajectory. This can be explained by the fact that the hands balance the robot to avoid falling backward. We also observe the compromise on accuracy that the robust controllers make: wherever the robust and trajectory-specific boxplots do not overlap, it means the robust controllers are not even in the top 50 for that trajectory.

IV. CONCLUSION

Hands-on time is saved by learning the controller parameters using our optimization approach, and in addition the results outperform hand-tuned parameter sets. In order to validate our results further, the next step is to test the learned controllers on the real Talos robot.

REFERENCES

[1] A. Rocchi, E. M. 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)*.

[2] L. Penco, E. M. Hoffman, V. Modugno, W. Gomes, J.-B. Mouret, and S. Ivaldi, "Learning robust task priorities and gains for control of redundant robots," *IEEE Robotics and Automation Letters*.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*.

[4] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer US.