Multiple Task Optimization using Dynamical Movement Primitives for Whole-Body Reactive Control

Ryan Lober[1], Vincent Padois[1] and Olivier Sigaud[1]

***Abstract*— Whole-body controllers provide the tools to execute multiple simultaneous tasks on humanoid robots, but given the robot's internal and external constraints, interferences are often generated which impede task completion. Priorities can be assigned to each task to manage these interferences, unfortunately, this is often done at the detriment of one or more tasks. In this paper we present a novel framework for defining and optimizing multiple tasks in order to resolve potential interferences prior to task execution and remove the need for prioritization. Our framework parameterizes tasks with Dynamical Movement Primitives, simulates and evaluates their execution, and optimizes their parameters based on a general compatibility principle, which is independent of the robot's topology, tasks or environment. Two test cases on a simulation of a humanoid robot are used to demonstrate the successful optimization of initially interfering tasks using this framework.**

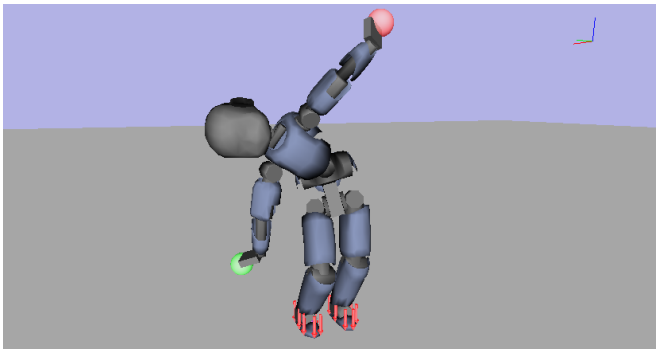

Fig. 1: A simulation snapshot of a humanoid robot attempting multiple challenging tasks simultaneously.

I. INTRODUCTION

Complex robotic topologies, such as humanoids, can provide versatile platforms with which to accomplish multiple tasks simultaneously. Yet, this versatility comes at the cost of ease of control. Modern control architectures, such as Task-Based Reactive Control (TBRC) formulate the control problem as a constrained minimization of task errors, and can efficiently exploit the redundancy of these platforms [1]. With the tools of TBRC, multiple tasks and constraints can be combined, and the resulting optimum represents the joint torque commands which would best accomplish this particular combination. Unfortunately, when multiple tasks are carried out simultaneously, it is often the case that the execution of one will perturb the execution of another. These interferences between simultaneous tasks can engender unpredictable behaviors.

Responding to this issue, a number of task prioritization architectures have been developed to manage these incompatibilities. Generally, these prioritization schemes are either weighted or hierarchical in nature. In weighted prioritization, the relative priorities between tasks are modulated by changing the weights associated with their errors and all tasks influence each other to some degree [1]. In hierarchical prioritization, priorities between tasks are strict and low priority tasks do not influence the higher priority tasks [2].

While necessary in many cases, such prioritization is often impossible or meaningless for some contexts. In these situations, one is concerned with simply completing all of the desired tasks without any emphasis on their priority. In this study, completing a task means attaining its desired goal state. While many tasks, such as welding, depend on the path taken from start to finish, tasks that do not enforce strict requirements on the trajectory, allow some flexibility throughout the duration of the task execution. Using this flexibility, the intermediate states of goal dependent tasks can be optimized to provide a set of tasks which will not perturb one another. Removing the interferences in a combination of tasks also removes the need to prioritize between the tasks, rendering them compatible. Informally, if the robot, on which multiple simultaneous tasks are applied, can successfully follow all task reference trajectories within an acceptable margin of error, then these tasks are compatible and do not interfere with one another.

In order to optimize tasks to render their combination compatible, we develop the Multiple Task Optimization Framework (MTOF). In the framework, parametric models known as Dynamical Movement Primitives (DMPs), first learn the tasks via regression techniques. The set of tasks, parameterized by DMPs, is then executed on a simulation of the humanoid robot in question, and information from the simulation is used to asses the compatibility of the tasks. In this study, the compatibility cost of a set of tasks is quantified by the error between the simulated execution of the tasks and their reference values. Based on this cost, the DMP parameters are modified via stochastic optimization to tend to minimize the successive costs. Once some minimum compatibility cost criterion has been met, a set of compatible tasks is output and can be passed to the real robot with equivalent priorities. Through the MTOF we are able to eliminate the need for task prioritization in TBRC by defining tasks

[1] The authors are with - Sorbonne Universités, UPMC Univ Paris 06, UMR 7222, Institut des Systèmes Intelligents et de Robotique, F-75005, Paris, France - CNRS, UMR 7222 , Institut des Systèmes Intelligents et de Robotique, F-75005, Paris, France
e-mail: firstname.lastname@isir.upmc.fr

with parameterized models, and optimizing those parameters based on a general compatibility principle.

In the following sections we present: a background of the tools and techniques used (Section II), a detailed overview of the MTOF (Section III), two experimental test cases with their results (Section IV), and concluding remarks (Section V).

II. BACKGROUND

In this section, a broad overview of the tools used in the development of the MTOF are presented as they pertain to this study.

A. Task-Based Reactive Control

To meet the challenges associated with whole-body humanoid control, convex optimization techniques have been used to develop reactive control architectures such as TBRC [3], [2], [4]. By formulating the control problem as an optimization problem, it is possible to avoid any numerically sensitive matrix inversions, directly incorporate equality and inequality constraints, assign a large number of tasks, and calculate a solution in the dynamic domain.

The equations of motion provide the relationship between the system dynamics and the joint-space variables,

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{g}(\boldsymbol{q}) = S\boldsymbol{\tau} + J_c^T(\boldsymbol{q})\boldsymbol{w}_e, \qquad (1)$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the joint-space positions, velocities and accelerations respectively. $M(\boldsymbol{q})\ddot{\boldsymbol{q}}$, $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{g}(\boldsymbol{q})$ represent the generalized inertia matrix, the non-linear effects (Coriolis and centrifugal) vector and the generalized gravitational force vector. The term, S, is an actuation matrix for the applied joint torques, $\boldsymbol{\tau}$, and accounts for unactuated joints. The contact Jacobian, J_c^T, projects the external contact wrenches, $\boldsymbol{w}_e$ into joint-space and provides their resultant joint torques. By concatenating $\ddot{\boldsymbol{q}}$, $\boldsymbol{w}_e$ and $\boldsymbol{\tau}$, $[\,\ddot{\boldsymbol{q}}^T, \boldsymbol{w}_e^T, \boldsymbol{\tau}^T\,]^T$, the dynamic variable, $\mathbb{X}$, is formed and allows the equation of motion, (1), to be represented in a compact form: $A\mathbb{X} = \boldsymbol{b}$ [1].

Given the dynamic relationship between operational-space and joint-space, $\ddot{\boldsymbol{x}} = J\ddot{\boldsymbol{q}} + \dot{J}\dot{\boldsymbol{q}}$, an acceleration task, T_i can be written,

$$T_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbb{X}) = \left\| \left(J_i(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}_i(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \ddot{\boldsymbol{x}}_i^* \right) \right\|^2 , \qquad (2)$$

where J_i and $\dot{J}_i$ are the task Jacobian and its derivative, and $\ddot{\boldsymbol{x}}_i^*$ is the desired operational-space acceleration to affect[1]. The task in (2) represents the squared error between the desired operational-space command and the actual joint-space accelerations of the system. The optimization problem can then be designed to find the minimum of the weighted sum of these squared errors, subject to the problem constraints,

$$\begin{aligned} \underset{\mathbb{X}}{\text{argmin}} \quad & \tfrac{1}{2}\textstyle\sum_{i=1}^{n_t} w_i T_i \\ \text{subject to:} \quad & G\mathbb{X} \preceq \boldsymbol{h} \\ & A\mathbb{X} = \boldsymbol{b}. \end{aligned} \qquad (3)$$

[1]Equation (2) shows an acceleration task, but it is also possible to develop wrench and force tasks in the dynamic domain [5].

Here, w_i is the weight, or importance, of each task and n_t is the total number of tasks. In addition to the equality constraint representing the equation of motion, inequality constraints such as articulation or actuator limits can also be incorporated directly in the resolution of the control problem and are represented by the criterion, $G\mathbb{X} \preceq \boldsymbol{h}$.

This problem is convex quadratic and can be solved using a Second-Order Cone Program (SOCP) [6], or a Linear Quadratic Program (LQP) [5], among others. The optimization of a weighted sum of tasks leads to a Pareto-optimal controller and varying these weights allows the exploration of the Pareto front [7]. In other words, varying the task weights leads to different overall behaviors; this fact is exploited to give tasks different levels of priority during execution. As an alternative to this weighted sum formulation of the TBRC, it is also possible to develop a hierarchical task scheme where the optimization problem is solved for each task individually, in order of their importance, and the constraint equations are updated with each resolution [2]. In this study, only the weighted sum formulation is used.

B. Dynamical Movement Primitives

DMPs were originally developed as tools to learn movements or trajectories from demonstration. They are based on point or limit cycle attractor landscapes augmented with a learned forcing term to guide the trajectory from start to goal, and typically some additional coupling terms to regulate their temporal evolution [8]. In the context of humanoid robotics, DMPs present an interesting tool with which to learn, modify and store various routine movements or tasks.

DMPs are constructed through the combination of one or more dynamical systems and a non-linear forcing term which allows their passage through space and time to be arbitrarily regulated. Following [9] and using the same scheme described in [10], the ensuing DMP is used in this work,

$$\begin{bmatrix} \dot{\boldsymbol{y}} \\ \dot{\boldsymbol{z}} \\ \dot{\boldsymbol{y}}^\gamma \\ \dot{\chi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}/\tau \\ ((K(\boldsymbol{y}^\gamma - \boldsymbol{y}) - D\boldsymbol{z}) + v \cdot \boldsymbol{f}(\chi, \boldsymbol{\theta}))/\tau \\ -\alpha_\gamma(\boldsymbol{\gamma} - \boldsymbol{y}^\gamma) \\ 1/\tau \\ -\alpha_v v(1 - v/v_{max}) \end{bmatrix} \begin{matrix} (a) \\ (b) \\ (c) \\ (d) \\ (e) \end{matrix} . \qquad (4)$$

The basic building block of this DMP is the spring-damper dynamical system, shown in lines (a) and (b), in which $[\,\boldsymbol{y}^T, \dot{\boldsymbol{y}}^T\,]^T$ is the state of the system, and $\ddot{\boldsymbol{y}} = \dot{\boldsymbol{z}}/\tau$ is the system output. The vector $\boldsymbol{y}$ typically contains the position and possibly orientation for some frame of reference. The terms K and D represent the stiffness and damping coefficients respectively, and are generally set for critical damping with $D = 2\sqrt{K}$. The attractor, or goal term, $\boldsymbol{\gamma}$, forces the system to converge to a specific point, and τ is proportional to the duration of the movement the DMP is approximating. Finally, the forcing term, $\boldsymbol{f}$, parameterizes the system's evolution from start to finish and is an open-loop controller which is a function of its parameters, $\boldsymbol{\theta}$, and an auxiliary dynamical system, χ.

To make sure the open-loop forcing term degrades to zero near $\boldsymbol{\gamma}$, the sigmoid function gating system, shown in line

(e), is used to regulate $\boldsymbol{f}$ where, α_v and v_{max}, are factors which regulate the decay rate and time of the system. The state χ is governed by the phase system[2], in line (d), which mimics the passage of time. The exponential goal system, in line (c), is also added to prevent high initial accelerations, and causes the virtual goal, $\boldsymbol{y}^{\gamma}$, to decay exponentially towards the true goal point, $\boldsymbol{\gamma}$.

The output of this DMP is a set of rates of change for the system states, which provide a trajectory controller in the DMP's parameter space. The reader is directed to [9], [8] and [10] for a more in depth discussion of each of these systems.

The forcing term, $\boldsymbol{f}$, is formulated as a non-linearly weighted combination of linear basis functions which allows it to approximate non-linear functions using non-linear regression techniques [11],

$$\boldsymbol{f}(\chi, \boldsymbol{\theta}) = \sum_i \psi_i(\chi)(a_i\chi + b_i). \tag{5}$$

Here the affine basis function $a_i\chi + b_i$ is weighted by a non-linear Gaussian,

$$\psi_i(\chi) = \exp\left(-\frac{1}{2\sigma_i^2}(\chi - c_i)^2\right). \tag{6}$$

In total, the forcing term is parametrized by four values: the affine slopes and offsets, $\boldsymbol{a}$ and $\boldsymbol{b}$, as well as the Gaussian centers and widths, $\boldsymbol{c}$ and $\boldsymbol{\sigma^2}$,

$$\boldsymbol{\theta} = \left[\boldsymbol{a}^T, \boldsymbol{b}^T, \boldsymbol{c}^T, \boldsymbol{\sigma^2}^T\right]^T. \tag{7}$$

These parameters can be adjusted to approximate any demonstrated or simulated trajectory via Linearly Weighted Regression (LWR) [12], or some variant of this technique [13]. Once the forcing term parameters have been learned, it is possible to pass them to a stochastic optimization algorithm in order to modify them based on some fitness criteria [14], [13].

C. Stochastic Optimization

Stochastic optimization is the process of sampling some mean input vector, $\boldsymbol{\theta}^{\mu}$, with a specific covariance, Σ, evaluating these samples, assigning them a fitness, J, then adjusting their mean, $\boldsymbol{\theta}^{\mu}_{\text{updated}}$, and covariance, Σ_{updated}, in the direction of the natural gradient of the fitness curve. This form of optimization requires only a fitness criterion for incremental updates and can therefore be categorized as a black-box optimization technique.

The update of the vector mean and covariance is one of the most critical aspects of stochastic optimization and a number of methods exist to this end. Parameter updates are generally accomplished with either gradient estimation or probability weighted averaging [15]. For this work, the Policy Improvement though Black-Box optimization algorithm, or PI^{BB}, is used [16], [13]. In PI^{BB}, the probability,

[2]This system is referred to as the "canonical" system in [8].

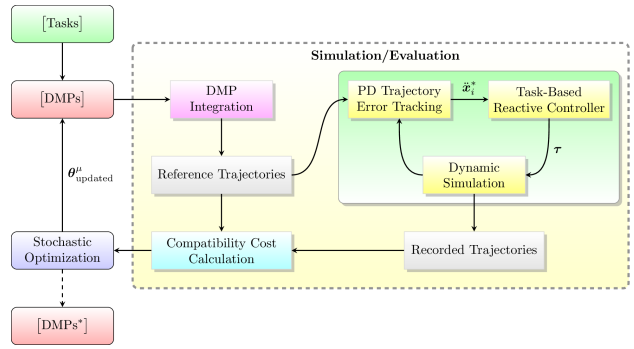

Fig. 2: High level flow chart of the MTOF showing a detail of the simulation and evaluation components. The integration of the dynamic simulator and the TBRC allow the MTOF to be applied to complex systems and controllers.

p_i, given to a particular sample, $\boldsymbol{\theta}^{\mu}_i$, is calculated as,

$$p_i = \exp\left(-h\frac{J_i - \texttt{min}(J)}{\texttt{max}(J) - \texttt{min}(J)}\right), \tag{8}$$

where h is an eliteness parameter which governs the tradeoff between exploration and exploitation. Using these probabilities, the vector mean update is calculated as,

$$\boldsymbol{\theta}^{\mu}_{\text{updated}} = \boldsymbol{p}^T\Theta, \tag{9}$$

where Θ is the concatenation of the sample vectors (candidate solutions) and $\boldsymbol{p}$ is the vector of their probabilities.

III. MULTIPLE TASK OPTIMIZATION FRAMEWORK

The first step in the MTOF, as shown in Fig. 2, is the parametrization of the original tasks by DMPs. The original tasks considered in this study are all operational-space tasks, but it is possible to parameterize joint-space tasks with DMPs. The forcing terms of each DMP are trained to approximate each task through LWR and provide parameterized models of the task DoFs. These forcing term parameters, $\boldsymbol{\theta}$, govern the evolution of the task states. The DMPs, representing the tasks, are then integrated to provide reference trajectories in operational-space. These reference trajectories are then passed to a proportional-derivative (PD) trajectory error tracking controller. This PD error tracking outputs desired accelerations, $\ddot{\boldsymbol{x}}_i^*$, for the reference frames associated with each task. The TBRC then solves the optimization problem for these commands with equal priorities between the primary tasks[3]. The output of the TBRC is a vector of actuation torques, $\boldsymbol{\tau}$. The movements generated by these torques are then simulated using the dynamic simulator and the trajectories followed by the task reference frames are recorded.

To determine the compatibility cost of a group of DMPs, the recorded trajectories from the iCub simulations are compared to the reference trajectories generated by the DMPs. The difference between the recorded and reference

[3]In TBRC, it is necessary to include a low weight joint-space torque task which ensures uniqueness of the solution [5], and this task is not taken into account here.

trajectories is then used as the cost function for the stochastic optimization updates,

$$J = \overbrace{\sum_{t=\text{start}}^{\text{end}} \frac{\epsilon(t)}{N_t}}^{\text{Trajectory Cost}} + \overbrace{w\epsilon(t_{\text{end}})}^{\text{Goal Cost}} \tag{10}$$
$$\epsilon(t) = \left\| \boldsymbol{x}^{\text{rec}}(t) - \boldsymbol{x}^{\text{ref}}(t) \right\|^2 .$$

Here J is the total compatibility cost, $\boldsymbol{x}^{\text{rec}}(t)$ and $\boldsymbol{x}^{\text{ref}}(t)$ are the position vectors of the recorded and reference trajectories at time t, and $\epsilon(t)$ is the squared norm of their difference at t. The term, $\epsilon(t_{\text{end}})$, is the squared norm of the difference between the end positions of the recorded and reference trajectories. The trajectory error cost is averaged over N_t, the number of time steps, and the goal cost is weighted by the term w.

The trajectory cost gives a measure of how much the robot must deviate from the reference trajectories in order to execute all of the tasks. The hypothesis here is that, if the robot cannot execute the tasks according to their reference trajectories, then some form of interference must exist. This criterion allows us to account for, and adjust to, the presence of interferences without explicitly modeling them.

Because we are primarily concerned with the tasks achieving their goal locations, the goal cost is added to this compatibility criterion to penalize task combinations which may have low tracking error for part of their execution but never attain their goals.

Using the costs provided by the simulation and evaluation stage, the stochastic optimization can update the parameters of the DMPs, which are concatenated to form its input vector, $\boldsymbol{\theta}^{\mu} = \left[\boldsymbol{\theta}_1^T, \boldsymbol{\theta}_2^T \ldots \boldsymbol{\theta}_{n_{\text{DMP}}}^T\right]^T$, to minimize the cost of their execution, or in other words, improve their compatibility. The reference trajectories, $\boldsymbol{x}^{\text{ref}}$, are therefore modified after each update to increase compatibility. After an update, the simulation, evaluation and optimization cycle is iterated until some compatibility criterion threshold is met. This threshold can be selected automatically based on an average observed tracking error during individual task execution. The optimized parameters form a set of compatible DMPs, which can be carried out on the real system without any form of prioritization.

IV. HUMANOID DEMONSTRATIONS

By integrating the TBRC developed in [5] as well as the robotic simulation software, XDE [17], [18], the MTOF can be demonstrated on a simulation of the iCub, a humanoid robot with 32 actuated DoF[4]. The iCub is simulated to be standing on a hard flat surface under its own power, and the contact forces are represented by arrows shown at the feet.

A. Test Case 1: Configuration Interferences

To test the MTOF, a set of three principle tasks is designated for the iCub. The first task is a standing task which maintains a frame of reference in the iCub's waist at a specific height at all times. The other two tasks are associated with reference frames in the palms of the iCub's left and right hands. The hand tasks are initially provided as straight line minimum jerk trajectories from the hands' starting positions to two goal points, indicated by the red and green spheres, in front of and behind the iCub, respectively in Fig. 3(a). The durations of the hand task movements are both equal to 4.0 seconds. All three tasks have equal priorities, 1.0, in the TBRC, and only the hand tasks are optimized in the MTOF.

[4]The real iCub robot has 18 hand DoFs and 3 camera DoFs that are not modeled in the simulation.

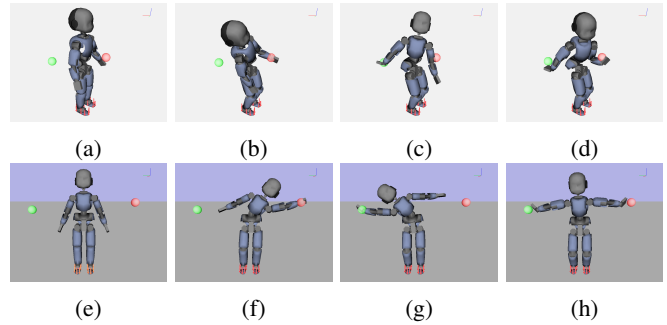

Fig. 3: Interference test cases. (a) - (d) show the configuration interferences and (e) - (h) show the workspace interferences.
(a) & (e) Starting position.
(b) & (f) Left hand + standing tasks end configuration.
(c) & (g) Right hand + standing tasks end configuration.
(d) & (h) End configuration of left hand, right hand and standing task combination where the robot cannot attain both goal positions (spheres) simultaneously.

B. Test Case 2: Workspace Interferences

Similarly to the first test case, three primary tasks associated with the two hands and waist are combined. The hand tasks are attached to reference frames in the palms of the left and right hands and are defined by two straight line minimum jerk trajectories to the sides of the iCub. The standing task, as before, simply maintains a reference frame in the iCub's waist at a fixed height at all times.

The hand goal positions are outside of the iCub's workspace and therefore the robot can never maintain both goals simultaneously. If both tasks end at the same time, then modification of the DMP forcing terms will not render the combination fully compatible because their attractor points are incompatible. Here, one must consider the temporal nature of the problem to find an appropriate resolution i.e. both tasks must be modified and done in some sequence, in order for the robot to meet its objectives. Therefore, the two hand trajectories have different durations, with the right hand trajectory being shorter than the left hand trajectory, 3.0 and 4.0 seconds respectively. To examine the effectiveness of the MTOF in cases where task sequencing is a factor, two examples are studied: 1) The task goal positions are maintained as attractor points even after the end of the duration of the task, therefore the task remains activated. 2) Once a hand task has reached the end of its duration and its goal position, within some small margin of error, the task is deactivated and no longer influences the resolution of

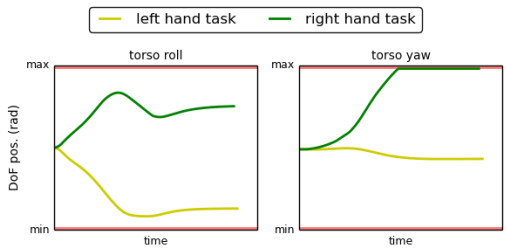

Fig. 4: Evolution of torso DoFs during the execution of the left hand + standing and right hand + standing combinations.

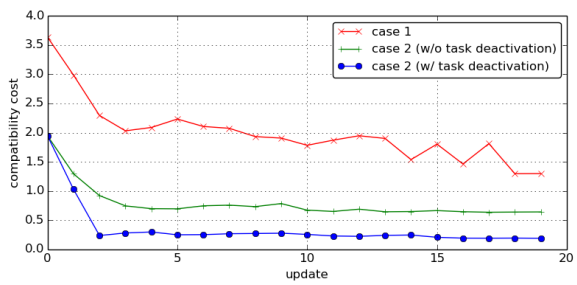

Fig. 5: Cost curves for the MTOF optimizations for both test cases.

the TBRC. All three tasks have equal priorities, 1.0, in the TBRC, and only the hand tasks are optimized in the MTOF.

C. Test Case 1: Results

When either the right or left hand task is combined with the standing task, the combination is feasible; see 3(b) and 3(c). However, the combination of all three generates interferences, 3(d). This is primarily due to the fact that each hand task generates opposing commands for the torso roll and yaw as seen in Fig. 4. The result of these interferences, shown in Fig. 3(d), is that the hand tasks never achieve their final goal positions and poorly track their reference trajectories.

Applying the MTOF to the configuration interferences case results in optimized tasks which reduce the compatibility cost from (10), and attain the final goal positions for the hand tasks. The evolution of the cost curve for the test cases is presented in Fig. 5 and a time-lapse of the movement generated by the optimized tasks is shown in Fig. 6.

D. Test Case 2: Results

In Figs. 3(f) and 3(g) it can be seen that, when either the left or the right hand tasks are executed with the standing task, the combinations are feasible. However, in Fig. 3(h) the combination of all three tasks generates workspace interferences caused by the fact that the robot's workspace is not large enough to accommodate both trajectories. This is illustrated in Fig. 7, where the distance between the tasks at each instant in time while they are active is plotted for the original and optimized sets of tasks. At approximately 2.1 seconds, the distance between the original tasks begins to exceed the maximum workspace of the iCub, and generates interferences. Since neither of the tasks reaches its goal, they

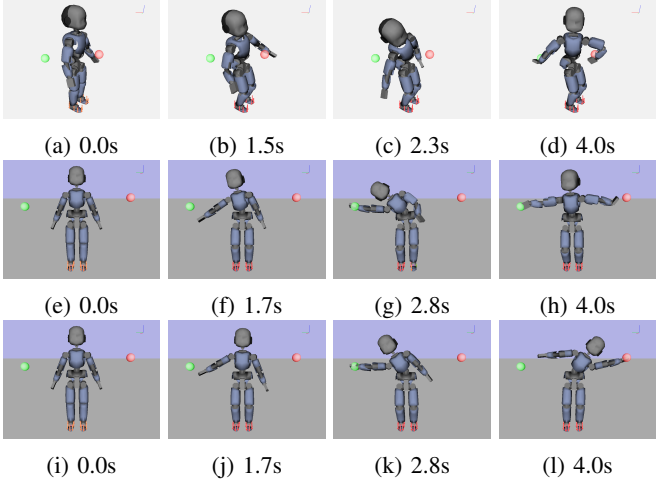

Fig. 6: Optimized task combinations found with MTOF. (a) - (d) show a time-lapse of the the optimized tasks for test case 1 and (e) - (l) show two time-lapses of the the optimized tasks for test case 2. (e) - (h) show the solution found without task deactivation and (i) - (l) show the solution found with task deactivation.

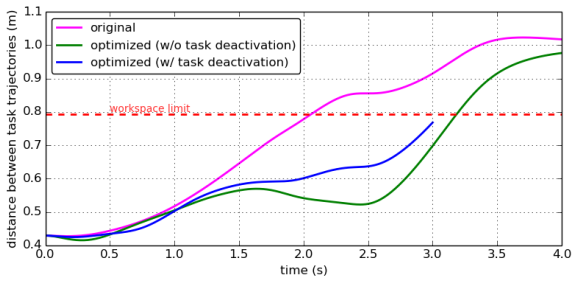

Fig. 7: A plot of the distances between the hand task trajectories for each instant in time when they are simultaneously active. The optimized tasks without task deactivation delay this interference, and the optimized tasks with task deactivation remove it.

cannot be deactivated and TBRC alone cannot accomplish this task combination.

1) Without Task Deactivation: The set of tasks optimized by the MTOF without task deactivation, modifies the hand trajectories to respect the workspace limit for as long as possible, and manages to attain the right hand task goal position. Unfortunately, because the right hand task is not deactivated, its goal position remains as a system attractor, and the robot is forced to violate the workspace limit. The overall result of this solution is a set of tasks which reduces the compatibility cost (see Fig. 5), but cannot attain all of the desired goal positions.

2) With Task Deactivation: When the tasks are optimized in the MTOF with task deactivation, the overall compatibility cost is reduced as shown in Fig. 5, and the hand tasks each attain their goal positions as shown in Figs. 6(k) and 6(l). This is possible because once the right hand goal position is attained, that task is deactivated and no longer contributes to the compatibility cost of the execution. This is reflected in

Fig. 7, where the distance between task curve terminates at 3.0 seconds, when the right hand task is completed.

V. CONCLUSIONS

Here, we have presented a novel framework, the MTOF, with which to modify a set of interfering tasks in order to render them compatible. Through test cases involving two general categories of interferences, it was shown that the MTOF was able to successfully modify the initially interfering tasks, via stochastic optimization of their parameters, in order to generate feasible combinations which could be carried out in TBRC without any form of prioritization. In this case, the TBRC has been formulated as a weighted sum of task errors (see (3)) and therefore the weights of the optimized tasks found in the test cases can all be set to 1.0 without generating interferences.

The general concepts behind MTOF are versatile and can be modified based on the user need. For example, here we are concerned with task compatibility and the removal of task prioritization, but using this same framework, one could easily integrate other optimization criteria related to energy consumption, distance from joint limits, bipedal stability, etc. Additionally, it is possible to optimize one, all, or some of the tasks, and any combination of their parameters, allowing flexibility in the problem resolution approach. Here, DMPs were used because of their goal convergence properties, but virtually any other parametric model could be used in its place.

The use of DMPs as parametric task representations allows the temporal nature of certain incompatibilities to be taken into account. The TBRC used as the basis for this work is a reactive controller which does not take into account any time horizon. The amalgam of these two components allows the MTOF to account for, and correct, interferences generated by the task combination in TBRC before they occur, effectively giving TBRC some very rudimentary form of pre-planning.

Although the MTOF can effectively optimize interfering tasks, its current major drawback is the computational cost of the task execution simulations. Because each update of the task parameters requires a number of simulations to be executed, the average time needed to optimize a set of tasks is too long for a real-world implementation. However, the MTOF can be done offline prior to task execution, so computational time is not critical and reducing the simulation time, as well as parallelization of the simulations, should improve on these shortcomings. In addition, the parameter updates can be incrementally improved, meaning that the tasks can be progressively optimized over the lifetime of the robot and not necessarily all at once.

The main takeaway from this study is that using the MTOF, we are able to solve a complex control problem using simple and general principles. Defining tasks with parameterized models allows them to be modified and optimized using black-box optimization tools, to meet any arbitrary criterion. Simulation and evaluation provide a means to incorporate the complex dynamics of the robot and its environment into the optimization criterion. Using these same methods and principles, one could easily apply the MTOF, or some variation of it, to other complex robotics problems.

ACKNOWLEDGMENTS

This work was partially supported by the European Commission, within the CoDyCo project (FP7-ICT-2011-9, No.600716) and by the RTE company through the RTE/UPMC chair Robotics Systems for field intervention in constrained environments held by Vincent Padois.

REFERENCES

[1] 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 (ICRA)*, 2011, pp. 1283–1290.

[2] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond, "Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 2939–2944.

[3] J. Park, J. Haan, and F. C. Park, "Convex optimization algorithms for active balancing of humanoid robots," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 817–822, 2007.

[4] L. Saab, "Generating whole body movements for dynamics anthropomorphic systems under constraints," Ph.D. dissertation, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2011.

[5] J. Salini, "Dynamic control for the task/posture coordination of humanoids: toward synthesis of complex activities," Ph.D. dissertation, Université Pierre & Marie Curie, 2012.

[6] L. Han, J. C. Trinkle, and Z. X. Li, "Grasp analysis as linear matrix inequality problems," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 663–674, 2000.

[7] T. W. Athan and P. Y. Papalambros, "A note on weighted criteria methods for compromise solutions in multi-objective optimization," *Engineering Optimization*, vol. 27, no. 2, pp. 155–176, 1996.

[8] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors." *Neural computation*, vol. 25, no. 2, pp. 328–73, Feb. 2013.

[9] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Wörgotter, Frgotter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2012.

[10] F. Stulp, "`DmpBbo` – a c++ library for black-box optimization of dynamical movement primitives." 2014. [Online]. Available: https://github.com/stulp/dmpbbo.git

[11] O. Sigaud, C. Salaun, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: a survey," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129, December 2011.

[12] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[13] T. Munzer, F. Stulp, and O. Sigaud, "Non-linear regression algorithms for motor skill acquisition: a comparison," *Proceedings JFPDA*, pp. 1–16, 2014.

[14] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud, "Learning compact parameterized skills with a single regression," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[15] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

[16] ——, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn. Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, September 2013.

[17] X. Merlhiot, J. L. Garrec, G. Saupin, and C. Andriot, "The xde mechanical kernel: Efficient and robust simulation of multibody dynamics with intermittent nonsmooth contacts," [Online]. Availible: http://www.kalisteo.fr/lsi/aucune/a-propos-de-xde, 2012.

[18] H. Sovannara. (2013, Dec) xde-isir wiki. [Online]. Available: http://pages.isir.upmc.fr/∼hak/xdewiki/doku.php?id=start