Efficient Reinforcement Learning for Humanoid Whole-Body Control

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

***Abstract*— Whole-body control of humanoid robots permits the execution of multiple simultaneous tasks but combining tasks can often result in unexpected overall behaviors. These discrepancies arise from a variety of internal and external factors and modeling them explicitly would be impractical. Reinforcement learning can be used to eliminate the effects of the deleterious factors through trial and error but generally requires many trials to converge on a solution. In humanoid robotics such improvidence can be costly. In this paper we show how the efficiency of the learning can be improved through use of Bayesian optimization. This is accomplished by intelligently exploring a model of the latent cost function derived from the quality of the task executions. We demonstrate the efficacy of the technique through two different simulated scenarios where various factors impede the robot from accomplishing its objectives.**

I. INTRODUCTION

Humanoids are versatile robotic platforms, which derive their utility from a high degree of redundancy and the ability to interact with their environment. Whole-body controllers exploit this by controlling all degrees of freedom (DoF) simultaneously [1], allowing multiple concurrent tasks to be executed under constraints. Task combinations and sequences can produce complex overall behaviors and are a major step towards real-world applications [2]. Unfortunately, determining the right task(s) for a certain job is a challenging problem [3]. Furthermore, the combination of multiple tasks often generates unexpected/unwanted overall behaviors. These discrepancies can arise from a number of sources, task priorities, loss of Jacobian rank, model/environment constraints and uncertainties, perturbations, etc. For simplicity we will refer to these sources as *disturbances*.

Considering the innumerable reasons for which a task may fail in a dynamic environment, at the planning stage, is computationally intractable and practically inefficient. One solution to this problem is to use reinforcement learning (RL) [4], [5], [6]. The robot executes its task(s), evaluates them according to one or multiple criteria, and then uses this metric to improve the task(s). This improvement is realized by optimizing the task parameters according to the metric, which in RL literature can be referred to as a reward, score, fitness, or cost — the term, cost, is used in this study. Given the task execution cost, the goal is to minimize this cost by adjusting the task parameters [7], [8]. Given the typically non-convex and non-linear nature of the cost functions, sample based optimization techniques, such as stochastic optimization, are commonly employed[1]. These stochastic optimization routines scale well with problem dimensionality but require many task executions to find an optimum, primarily due to the fact that many samples are needed to estimate the cost function gradient [9]. Compounding this, is the fact that the the mapping from the space of the optimization variables, or task parameters, to the space of the cost function is highly non-linear and possibly discontinuous [5].

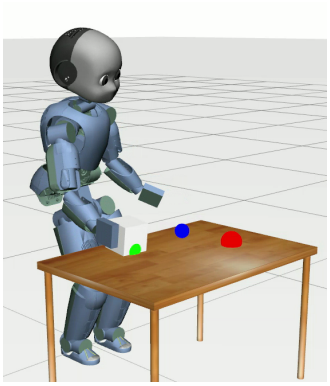


Fig. 1: The humanoid robot, iCub, performing a complex combination of tasks.

Executing many trials on a humanoid robot is a time-consuming and possibly dangerous for the robot. Additionally, because stochastic search is unbounded, one has to be careful that the robot does not test overly aggressive and possibly harmful movements during the RL — especially when balancing is involved.

The intent of this study is to provide a straightforward way of improving on these shortcomings. To do so, we present two contributions. Firstly, in Section III-B, we propose a way of evaluating how well a task was executed and indirectly measuring the influence of unpredicted disturbances. This metric is represented by a set of cost functions which enable the appraisal of the *quality* of task execution. Secondly, in Section III-C, we show that by choosing a more intuitive and low-dimensional task parameterization, we are able to use Bayesian optimization (BO), which has proven to be the technique of choice for optimizing expensive cost functions in few trials [10], as is the case with robotic experimentation [11], [12], [13], [14]. The key benefit of BO is that it models the underlying cost function given a few samples and proposes the next best set of parameters to test which will both minimize the predicted cost and improve the cost function estimation. This procedure quickly and efficiently explores the parameter space for the tasks in a bounded manner.

II. BACKGROUND

In the following section, we provide the requisite background information for this work. An overview of task-based whole-body control is presented, followed by a brief introduction to BO.

[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

[1]This is typically referred to as policy search.

A. Whole-Body Control

Whole-body controllers reactively calculate the joint torques necessary to accomplish some set of tasks in either operational-space or joint-space, for all of the DoF of the given robot, while respecting physical constraints. This is done by solving a constrained convex optimization problem at each time step. The objective of the problem is a weighted sum of task errors, and its constraints are the equations of motion, articulation and actuation limits, and contacts. Task errors, are calculated as the difference between the current system accelerations/wrenches and some desired values. These desired values, are generally provided by trajectories. The control problem can then be solved efficiently using a Quadratic Program. The reader is directed to [2] for a more detailed explanation of this controller. The weight associated with each task error determines its relative priority, with larger weights meaning higher priorities. Alternatively, hierarchical approaches to task priorities can be implemented [15], [16].

B. Bayesian Optimization

BO is a non-linear and non-convex optimization technique, popular in fields where evaluating a cost function is burdensome, as in robotics. As with any optimization, the goal of BO is to minimize a cost function, f,

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} f(\boldsymbol{\omega}) \,, \tag{1}$$

with optimization variables, $\boldsymbol{\omega}$. Unlike other optimization techniques, rather than randomly sample the cost function to estimate gradient information, BO constructs a global model of the cost function based on the observed samples $\left[\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \dots \boldsymbol{\omega}_n\right] \in \Omega$ and their corresponding costs, $\left[j_1, j_2, \dots j_n\right] \in \boldsymbol{j}$. This model, commonly referred to as the surrogate function, is then used to determine the next set of parameters to test, which both minimize the estimated cost and improve the model accuracy.

The tool used to build this model is known as a Gaussian process (GP). For an unobserved optimization variable set, $\hat{\boldsymbol{\omega}}$, it tells us their predicted cost mean, $\hat{j}_\mu$ and variance, $\hat{j}_{\sigma^2}$[2].

$$\begin{bmatrix} \hat{j}_\mu \\ \hat{j}_{\sigma^2} \end{bmatrix} = \begin{bmatrix} K_* K^{-1} \boldsymbol{j}^T \\ K_{**} - K_* K^{-1} K_*^T \end{bmatrix} = \mathcal{GP}(\hat{\boldsymbol{\omega}}, \Omega, \boldsymbol{j}) \,. \tag{2}$$

In (2), the K matrices are determined by evaluating a mean and covariance function over the training and unobserved data. Here we use the zero mean and squared exponential covariance kernel functions [17]. The term K is the projection of each training datum, $\boldsymbol{\omega}_i$, into the kernel space, K_* is the projection of $\hat{\boldsymbol{\omega}}$ into the kernel space, and K_{**} is the projection of $\hat{\boldsymbol{\omega}}$ into a kernel defined by itself. Evaluating (2) is commonly referred to as GP regression and implementation details can be found in [17].

Given the surrogate function, the next step in BO is to determine the optimal set of parameters, $\hat{\boldsymbol{\omega}}^*$, to sample on the next evaluation of the cost function. To do so, the exploitation and exploration of the surrogate function must be balanced. Simply choosing the minimum predicted value may leave us in a local minimum, and areas of high variance should be explored to improve the surrogate's accuracy. An acquisition function is therefore used to balance the exploration and exploitation of the knowledge about the cost function [18]. It is a function of the estimated cost means and variances, $f_{acq}(\hat{\boldsymbol{j}}_\mu, \hat{\boldsymbol{j}}_{\sigma^2})$. The terms $\hat{\boldsymbol{j}}_\mu$ and $\hat{\boldsymbol{j}}_{\sigma^2}$ are determined by evaluating the GP surrogate function over the entire input space, $\hat{\Omega}$,

$$\begin{bmatrix} \hat{\boldsymbol{j}}_\mu \\ \hat{\boldsymbol{j}}_{\sigma^2} \end{bmatrix} = \mathcal{GP}(\hat{\Omega}, \Omega, \boldsymbol{j}) \,. \tag{3}$$

To obtain $\hat{\Omega}$, the search space must be bounded and discretized. This means that the search space grows as m^n, where m is the number of increments between bounds, and n is the dimension of $\boldsymbol{\omega} \in \mathbb{R}^{n \times 1}$. This is the principle limitation of BO. Recently, a method for circumventing this limitation has been proposed in [19], but is not used in this work.

Minimizing f_{acq} w.r.t. $\boldsymbol{\omega}$, we find the next best set of optimization variables to test, $\hat{\boldsymbol{\omega}}^*$, which balance the exploration and exploitation of f,

$$\hat{\boldsymbol{\omega}}^* = \arg\min_{\boldsymbol{\omega}} f_{acq}(\hat{\boldsymbol{j}}_\mu, \hat{\boldsymbol{j}}_{\sigma^2}) \,. \tag{4}$$

The real cost function of our problem is sampled by evaluating the optimal variable set, $\hat{\boldsymbol{\omega}}^*$. In robotics, this corresponds to executing the task(s) and calculating their cost. Given this new data, $\boldsymbol{\omega}^*$, and its cost, j^*, the surrogate is updated[3]. The update consists simply of concatenating the new data to the old data and reevaluating (2). The optimization concludes when some problem-dependent convergence criterion is met.

III. METHODS

This section details the technical contributions of this work. We first detail waypoint-based task trajectories used to guide the evolution of tasks. Then we develop a general way of measuring execution quality of tasks. We proceed with the specific BO components necessary to optimize waypoint-based trajectories. Finally, the complete task optimization algorithm is presented.

A. Task Trajectories

Tasks in whole-body control are generally guided by some form of trajectory to ensure smooth movements. In this study, trajectories are generated from a series of waypoints, which represent coordinates of particular importance. Here, we use the variable, $\boldsymbol{\xi}$, to represent generic task trajectory coordinates. Inherent in these waypoints are the overall objectives of some task; e.g., for a point to point reaching task, the waypoints are given by the start and goal coordinates of the reach. A single waypoint, $\boldsymbol{\lambda}_i$, with n DoF, is given by,

$$\boldsymbol{\lambda}_i = \left[\xi_1, \xi_2, \dots, \xi_n\right]_i^T \,, \tag{5}$$

while a set of n_λ waypoints is denoted,

$$\Lambda = \left[\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_{n_\lambda}\right] \,. \tag{6}$$

A time step, t_i^λ, must be associated to each $\boldsymbol{\lambda}_i$. This can be done by hand or via some heuristic method. From Λ and $\boldsymbol{t}^\lambda$, a trajectory can be generated which outputs $\left[\boldsymbol{\xi}^T, \dot{\boldsymbol{\xi}}^T, \ddot{\boldsymbol{\xi}}^T\right]^T$ at each time step, t, from t_1^λ to $t_{n_\lambda}^\lambda$. In this study, the trajectory generation scheme proposed in [20] is used, which is tantamount to basis-spline functions.

[2]Henceforth, the hat symbol, $\hat{\bullet}$, is used to indicate and unobserved, or unmeasured, variable.

[3]Note the omission of the hat symbol, $\hat{\bullet}$, indicating that $\boldsymbol{\omega}^*$ and j^* have been observed.

B. Estimating Task Quality

To quantify the quality of a set of tasks in a whole-body controller, three component costs are calculated over the execution of the tasks. These costs provide an indirect measure of possible disturbances without explicitly modeling them.

Firstly we make the hypothesis that, if the trajectory tracking is imperfect, then for some reason disturbances exist, and the execution of the tasks in question is less than optimal. This is characterized by the tracking error cost,

$$j_{tracking}(t) = \|\boldsymbol{\epsilon}^{\boldsymbol{\xi}}(t)\|^2 \,, \tag{7}$$

where $\boldsymbol{\epsilon}^{\boldsymbol{\xi}}(t)$, is the task error w.r.t. its reference at time, t. Secondly it is assumed that the overall objective of any trajectory is to reach its goal coordinate, which is the final waypoint, i.e.

$$j_{goal}(t) = \gamma(t)\|\boldsymbol{\epsilon}^{\boldsymbol{\xi}}_{\boldsymbol{\lambda}_{n_\lambda}}(t)\|^2 \,, \tag{8}$$

where $\boldsymbol{\epsilon}^{\boldsymbol{\xi}}_{\boldsymbol{\lambda}_{n_\lambda}}(t)$, is the difference between the task's current coordinates and the final waypoint in its trajectory. The term $\boldsymbol{\epsilon}^{\boldsymbol{\xi}}$ is used to represent a generic error term and is not explicitly defined because it's calculation a function of the parameterization used (see Ch. 1 of [21]). The term $\gamma(t)$ is a function which weights the error over time, applying exponentially greater penalties as t exceeds $t^{\lambda}_{n_\lambda}$, i.e. $\gamma(t) = \left(\frac{t}{t^{\lambda}_{n_\lambda}}\right)^{10}$. Finally, to avoid energy inefficient tasks, an energy cost is calculated using,

$$j_{energy}(t) = \|\boldsymbol{\tau}(t)\|^2 \,. \tag{9}$$

Because j_{energy} is often orders of magnitude greater than the other component costs, it is necessary to scale this term. To do so, we take the maximum value of $\boldsymbol{j}^1_{energy}$ in the first execution of the tasks and use this as a scaling factor, i.e.

$$\boldsymbol{j}_{energy} := \frac{\boldsymbol{j}_{energy}}{\max(\boldsymbol{j}^1_{energy})} \,. \tag{10}$$

All subsequent task execution energy costs are scaled by the $\max(\boldsymbol{j}^1_{energy})$ factor from the first execution. The total quality cost, j_T, of the tasks is the time averaged sum of (7), (8) and (9),

$$j_T = \frac{\sum_{t=0.0}^{t_e} \left[j_{tracking}(t) + j_{goal}(t) + j_{energy}(t)\right]}{t_e} \,, \tag{11}$$

where t_e is the total duration of the task execution. In practice, we find that scaling j_T for each trial w.r.t. the j_T from the first iteration, similarly to the energy scaling in (10), is an effective technique for improving solver convergence and is implemented in this study, i.e.

$$j_T(i) := \frac{j_T(i)}{j_T(1)} \,. \tag{12}$$

C. Bayesian Task Optimization

Similarly to [22], we implement a Lower Confidence Bounds, or LCB, acquisition function because we are concerned with minimization.

$$LCB(\hat{\Omega}, \Omega, \boldsymbol{j}) = \hat{\boldsymbol{j}}_{\mu} - \sqrt{\kappa}\hat{\boldsymbol{j}}_{\sigma^2} \,, \tag{13}$$

where the term κ is calculated using a variant of the no-regret formulation developed in [23], $\kappa = 2\log\left(0.3i^{\frac{5}{2}}\pi^2\right)$. To determine the convergence of BO, we use the confidence, $\mathcal{C}$, of the optimal variable's predicted cost,

$$\mathcal{C} = \left(1 - \hat{j}^*_{\sigma^2}\right) 100 \,, \tag{14}$$

which yields a percentage measure of how sure we are that the optimal variables, $\hat{\boldsymbol{\omega}}^*$, will produce a cost, $\hat{j}^*_{\mu}$, if executed. Once $\mathcal{C}$ is higher than some threshold set by the user the optimization is stopped. This is a valid condition for convergence because the optimization occurs over the entire bounded and discrete search space. Thus any optimum found with sufficient confidence is a global optimum [10]. Using $\mathcal{C}$ for our convergence criterion eschews the need to choose a minimum change in cost for convergence. This is convenient because such criteria are often arbitrary or non-trivial to calculate.

The optimization variables, $\boldsymbol{\omega}$, must be selected from the trajectory waypoint data, Λ and $\boldsymbol{t}^{\lambda}$, by the user. For instance, consider the following trajectory waypoints,

$$\begin{array}{rcl} \Lambda & = & \begin{bmatrix}\boldsymbol{\lambda}_1 & \boldsymbol{\lambda}_2 & \boldsymbol{\lambda}_3\end{bmatrix} \\ \boldsymbol{t}^{\lambda} & = & \begin{bmatrix}t^{\lambda}_1 & t^{\lambda}_2 & t^{\lambda}_3\end{bmatrix} \,, \end{array} \tag{15}$$

depending on the meaning of these waypoints, we may not wish to modify them. However, for waypoints which do not represent strict goal coordinates for the task, we may have the liberty to modify their values and times. In the case of (15), we could chose to optimize the vector $\boldsymbol{\omega} = [\boldsymbol{\lambda}^T_2, t_2]^T$. Alternatively we could add an "optimization" waypoint to the set, $\boldsymbol{\omega} = [\boldsymbol{\xi}'^T, t']^T$, which could be picked from the trajectory at time, t',

$$\begin{array}{rcl} \Lambda & = & \begin{bmatrix}\boldsymbol{\lambda}_1 & \boldsymbol{\lambda}_2 & \boldsymbol{\xi}' & \boldsymbol{\lambda}_3\end{bmatrix} \\ \boldsymbol{t}^{\lambda} & = & \begin{bmatrix}t^{\lambda}_1 & t^{\lambda}_2 & t' & t^{\lambda}_3\end{bmatrix} \,. \end{array} \tag{16}$$

To bound the BO search space we find that the minimum and maximum values for each DoF of Λ and $\boldsymbol{t}^{\lambda}$,

$$\hat{\Omega} \in \left[\min\left(\begin{bmatrix}\Lambda \\ \boldsymbol{t}^{\lambda}\end{bmatrix}\right), \max\left(\begin{bmatrix}\Lambda \\ \boldsymbol{t}^{\lambda}\end{bmatrix}\right)\right] \,, \tag{17}$$

provide a good rule-of-thumb first choice. The space between these bounds is then discretized. The grid size used for discretization depends on the coordinate systems being used. For Cartesian waypoints, we estimate that our robot is no more accurate than 2cm so we use a 3cm grid size. For the time coordinates, we use some fraction of the movement duration e.g. $\frac{t_{n_\lambda}}{n_g}$, where n_g is the number of desired grid steps along $\boldsymbol{t}^{\lambda}$. Finally, the search space is scaled to $[0, 1]$ in each dimension to improve the cost function approximation. This is a common technique in machine learning known as feature scaling.

The overall whole-body task optimization framework is summarized by Algorithm 1. Given a set of tasks, their component costs, (7), (8), and (9) are computed during execution. Once finished, j_T is evaluated using (11). If the quality cost indicates a failure of the robot to achieve its goals, BO is initialized with the waypoint variables selected for $\boldsymbol{\omega}$ and their resultant j_T. The BO then proposes a new set of waypoint variables to test, $\hat{\boldsymbol{\omega}}^*$, which should decrease the cost of the task and improve the approximation of the latent cost function. The optimization waypoint variables, $\boldsymbol{\omega}$, are then replaced with $\hat{\boldsymbol{\omega}}^*$ and the tasks are executed on the robot. Using the quality cost, j^*_T, of the new variables, the BO update can be performed. This optimization loop is iterated until the BO confidence, $\mathcal{C}$, is greater than 99%.

Algorithm 1 Task Optimization Algorithm

given Λ and $\boldsymbol{t}^\lambda$, generate trajectory
execute tasks
calculate quality cost j_T (11)
select optimization waypoint data $\boldsymbol{\omega}$ (15) & (16)
determine optimization bounds (17)
while $\mathcal{C} < 99\%$ **do** (14)
  update BO with $\boldsymbol{\omega}$ and j_T
  solve for $\hat{\boldsymbol{\omega}}^*$ and $\mathcal{C}$ (4)
  $\boldsymbol{\omega} := \hat{\boldsymbol{\omega}}^*$
  execute tasks
  calculate quality cost j_T
end while

IV. SIMULATIONS

In this section, we detail the experimental setups of two simulated scenarios. The objective of these tests is to show how a variety of complex factors can degrade the overall behavior of the robot, and by optimizing for the task quality, we are able to correct for these factors. All simulations are executed in the Gazebo [24] environment using the ODE physics engine and a real-time clock[4]. A simulation of the humanoid robot, iCub, is used as the test platform. The first scenario provides a demonstration of the overall task optimization framework using a toy obstacle avoidance scenario. In the second, more complex, scenario a set of dynamically incompatible tasks, actuator limits, and environmental interactions generate a complicated set of factors which degrade the robot's ability to perform its desired tasks. In both tests, we explicitly prevent the use of environmental information for the RL. The code for this study is open access and can be downloaded from: `https://github.com/rlober/humanoids-2016`.

A. Obstacle Avoidance

In this toy example, the goal is to illustrate the key components of the task optimization. The root link of the robot is fixed to isolate task disturbances to the presence of the obstacle and remove any possible issues due to maintaining balance. The robot is given a trajectory for its right hand Cartesian task from the hand's starting position to a point 25cm above it with the following waypoints,

$$\begin{array}{rcl} \Lambda & = & \begin{bmatrix} z_{start} & z_{goal} \end{bmatrix} \\ \boldsymbol{t}^\lambda & = & \begin{bmatrix} t_{start} & t_{goal} \end{bmatrix}, \end{array} \tag{18}$$

where $z_{start} = 0.0$m, $z_{goal} = 0.25$m, $t_{start} = 0.0$s, and $t_{goal} = 2.5$s. The x and y axes of the task frame are constrained to force the robot to move in a straight line along the z-axis preventing the robot from moving around the obstacle. Full-body and torso posture tasks are used to keep the robot in an upright posture and avoid unactuated limbs.

The robot must optimize the right hand Cartesian task to avoid an obstacle, of which it has no knowledge or perception. Furthermore, because we cannot modify the right hand task waypoints, an optimization waypoint is inserted into (18),

$$\begin{array}{rcl} \Lambda & = & \begin{bmatrix} 0.0 & z' & 0.25 \end{bmatrix} \\ \boldsymbol{t}^\lambda & = & \begin{bmatrix} 0.0 & t' & 2.5 \end{bmatrix}. \end{array} \tag{19}$$

Here, t' is taken as the median time of the movement, 1.25s, and $z' = \xi(t') = 0.12$m. This waypoint is then used as the initial optimization variable data for Algorithm 1, $\boldsymbol{\omega} = [z', t']^T = [0.12, 1.25]^T$. The objective variable, $\boldsymbol{\omega}$, is in $\mathbb{R}^{2\times 1}$, so the surrogate function mean within the BO may be visualized. The task execution is terminated if the hand has reached its goal location to within 3.0cm or the movement has exceeded 10.0s in total duration. At exactly 1.0s, an obstacle is inserted into the path of the robot's hand trajectory at $z = 0.12$m; however, this information is not used in any way during the task optimization. This obstacle is a 1.0cm thick flat plate.

B. Moving a Heavy Weight

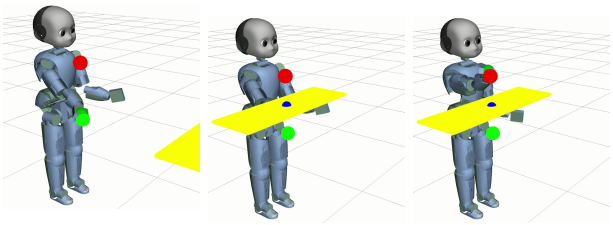
The aim of this simulation is to show how the task optimization can compensate for a complex combination of factors which impair the overall behavior of the robot. In this example, the robot is standing on a flat surface and maintaining balance, while trying to move a 1.0kg cube from its starting point to a target location on a flat table. The feet are not fixed to the ground. The cube is affixed to the right hand and its mass is greater than the maximum effective payload of the iCub's right arm, which is approximately 0.5kg when fully extended. An orientation task is used to maintain the cube's bottom surface parallel to the table top. Balance is achieved through a Center of Mass (CoM) position task, the objective of which is to maintain the CoM's ground projection in the center of the support polygon defined by the convex hull of feet contacts. A torso Cartesian task along with a full-body posture task keep the robot upright. Qualitatively, the principle sources of task quality degradation are the heavy payload, which the actuators of the right arm are incapable of correctly supporting, and the conflict between the right hand task and the overall balance of the robot. Task execution is terminated if the cube has reached its goal location to within 3.0cm or the movement has exceeded 15.0s in total duration.

The right hand trajectory consists of three waypoints, at the start, middle, and goal of the movement, $\Lambda = [\boldsymbol{\lambda}_{start}, \boldsymbol{\lambda}_{mid}, \boldsymbol{\lambda}_{goal}]$, and $\boldsymbol{t}^\lambda = [t_{start}, t_{mid}, t_{goal}]$. The middle waypoint is chosen at 5.0cm above the table to avoid dragging the cube, and is also selected as the optimization variable, $\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\lambda}_{mid}^T, t_{mid} \end{bmatrix}^T$. The optimization bounds are restricted in this simulation, to a hyperrectangle centered at $\boldsymbol{\omega}$. The bounds are given by $\boldsymbol{\omega} \pm [1.0, 0.05, 0.05, 0.02]^T$. We restrict the search space in this example to prevent the robot from exploring movements which would cause it to fall, to more accurately mimic a real-world reinforcement learning experiment.

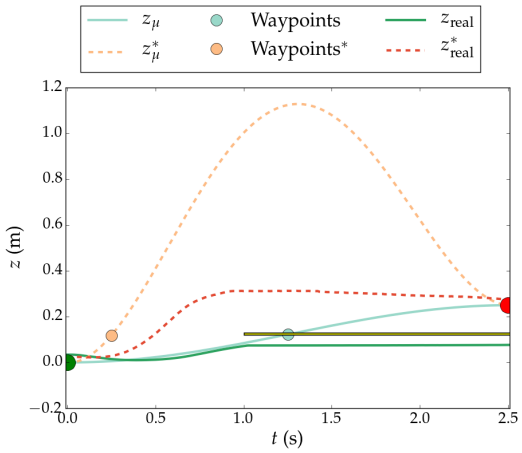
V. RESULTS

This section presents the results for the test simulations described in Sec. IV. For both tests, the time of an optimization iteration is equal to the time it takes to complete an execution, which is limited in both simulations to 10s and 15s respectively, plus the time it takes to compute the BO update, which never exceeds 1.0s. The BO update occurs in parallel to the task executions and takes far less time to compute than the robot does

[4]All simulations are run on a computer with an Intel® Core™ i7-4900MQ CPU, with a 2.80GHz clock.



(a) Starting position. (b) Original task. (c) Optimized task.



(d) Trajectory plots.

Fig. 2: This figure shows the original, z, and optimized, z^* tasks for the "Obstacle Avoidance" simulation detailed in Sec. IV-A. In 2(b) it can be seen that the robot is blocked by the yellow obstacle introduced at 1.0s. Task optimization generates a trajectory which causes greater acceleration at the beginning of the movement, moving the right arm above the obstacle before it arrives 2(c). This allows the robot to reach its goal, indicated by the red sphere, which we can confirm in the trajectory plot, 2(d).

to reset itself for another trial. Recordings of these simulations can be found in the submission's accompanying video.

A. Obstacle Avoidance

In Fig. 2, the main results of the simulation are presented. Figures 2(b) and 2(d) show that the original task execution fails due to the yellow obstacle impeding the movement of the hand. After 7 optimization iterations, it can be seen in Figs. 2(c) and 2(d) that the optimized trajectory moves the hand quickly enough to avoid the obstacle and attain its goal. One of the key things to note in Fig. 2(d) is that the optimal waypoint found for this task actually generates a trajectory outside of the robot's workspace. This has the effect of creating strong accelerations in the beginning of the movement allowing the robot to avoid the obstacle. Normally such a trajectory would generate dangerous movements, but the whole-body controller does not execute the infeasible portions of the trajectory because they violate the control constraints and conflict with the other tasks. This can be seen in Fig. 2(d) where the real task trajectory, z^*_{real}, only partially follows the reference, z^*_μ. This is interesting because the optimized trajectory is more counterintuitive than the original trajectory, yet it is an effective solution to this problem.

In Fig. 3, the BO surrogate function mean, $\hat{\boldsymbol{j}}_\mu$, is plotted. Here,

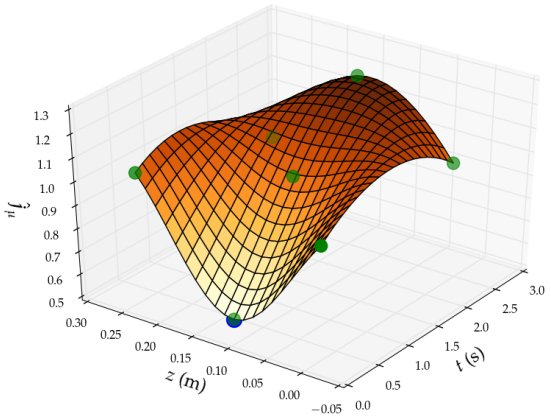


Fig. 3: A plot of the surrogate function cost mean, $\hat{\boldsymbol{j}}_\mu$, within the task optimization described in Algorithm 1. This figure shows the quality cost landscape of the obstacle avoidance simulation after 7 iterations. The task executions, or cost function samples, are plotted by the green dots. The blue dot represents the optimal variables found for this simulation. Using these values for the task's optimization waypoint, the obstacle is successfully avoided.

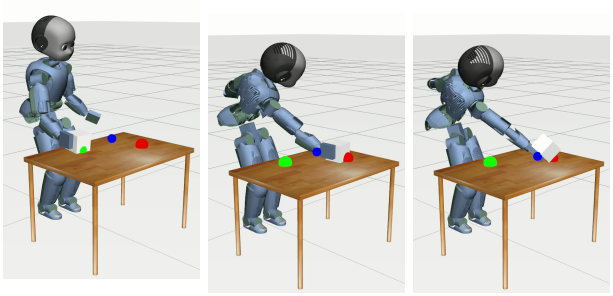
we observe that the cost landscape generated by this disturbance scenario is most elevated after the obstacle insertion time and below the z-height of the obstacle. This is logical because any waypoint in this region of the search space would result in a failure to reach the goal. Each sample of the real cost function is indicated by a green dot, and represents a task execution. The blue dot shows the optimal value found by BO with high confidence and corresponds to the optimal waypoint shown in Fig. 2(d).

B. Moving a Heavy Weight

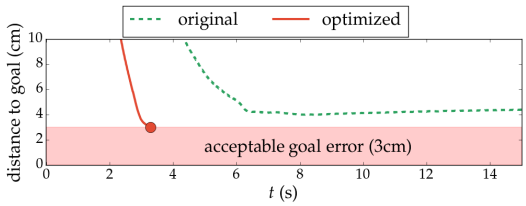
In Fig. 4, the original and optimized task executions for this example are shown. Although the differences are imperceptible in these still frames, the robot manages to attain the desired weight target to within 3.0cm of error, as shown by Fig. 4(d). After 11 task trials, the robot learns how to accomplish its objectives by placing the middle waypoint closer to the goal and sooner in the trajectory resulting in a task trajectory with higher initial accelerations. These accelerations increase the task error term and consequently the optimal torques needed to accomplish it. This causes the whole-body controller to preferentially accomplish the hand task and deviate the CoM task. Task equilibrium is reestablished at the end of the hand movement when the velocities and accelerations go to zero. This solution is quite intuitive to anyone who has moved a heavy object; however, it would be difficult to plan such a dynamic movement.

VI. CONCLUSION

Whole-body control can exploit the utility of humanoid robots to the fullest through multi-tasking; however, there is no guarantee that they will be executed properly due to task priorities, model/environment constraints and uncertainties, perturbations, etc. Accounting for these factors directly would be computationally untenable, and in this article, we showed how RL can be used correct such issues in an efficient manner.



(a) Starting position. (b) Original task. (c) Optimized task.



(d) Distance to goal.

Fig. 4: This figure shows the original and optimized tasks for the "Moving a Heavy Weight" simulation detailed in Sec. IV-B. In 2(c) it can be seen that the robot comes up just short of its goal location. By optimizing the middle waypoint of the task, the robot is able to accelerate more at the beginning of the movement, building up sufficient inertia to reach its goal within the acceptable error, as demonstrated by Fig. 4(d).

First it was demonstrated how to calculate the quality of task execution through a few simple cost functions. This metric measured how various disturbances affected the task executions without explicitly modeling them. Second by using the task trajectory waypoints as parameters, it was shown how BO could minimize the task quality cost and compensate for unexpected/unwanted behaviors. Via two different scenarios, the efficacy of this technique was demonstrated by solving complex problems in very few trials. We believe this to be a promising step towards more robust and effective RL on systems with little margin for error, such as humanoids.

Unfortunately, Bayesian optimization suffers from the curse of dimensionality and beyond 8 optimization variables we will quickly run out of memory. Therefore, the first course of action is to explore ways of eliminating the need to discretize the BO input space and use local search techniques instead. This would enable the method to be extended to higher dimensional tasks, such as those in the joint-space.

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] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 29–43, 2004.

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

[3] F. Keith, N. Mansard, S. Miossec, and A. Kheddar, "Optimization of tasks warping and scheduling for smooth sequencing of robotic actions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1609–1614.

[4] M. Mühlig, M. Gienger, S. Hellbach, J. J. Steil, and C. Goerick, "Task-level imitation learning using variance-based movement optimization," in *IEEE/RSJ International Conference on Robotics and Automation*, May 2009, pp. 1177–1184.

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

[6] S. Calinon, D. Bruno, and D. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *IEEE International Conference on Robotics and Automation*, May 2014.

[7] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement Learning in Robotics: A Survey," *The International Journal of Robotics Research*, 2013.

[8] F. Stulp and O. Sigaud, "Robot Skill Learning: From Reinforcement Learning to Evolution Strategies," *Paladyn Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, Aug. 2013.

[9] ——, "Policy improvement: Between black-box optimization and episodic reinforcement learning," pp. 1–15, 2012.

[10] J. Mockus, *Bayesian approach to global optimization: theory and applications*. Springer Science & Business Media, 2012, vol. 37.

[11] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, "Automatic Gait Optimization with Gaussian Process Regression," in *IJCAI*, vol. 7, 2007, pp. 944–949.

[12] Cully, A., Clune, J., Tarapore, D., and Mouret J.-B., "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, may 2015.

[13] R. Martinez-Cantin, N. de Freitas, and J. Castellanos, "Analysis of Particle Methods for Simultaneous Robot Localization and Mapping and a New Algorithm: Marginal-SLAM," in *IEEE International Conference on Robotics and Automation*, April 2007, pp. 2415–2420.

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

[15] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, 2014.

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

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

[18] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *ArXiv e-prints*, Dec. 2010.

[19] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams, "Scalable Bayesian Optimization Using Deep Neural Networks," *ArXiv e-prints*, Feb. 2015.

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

[21] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer Science & Business Media, 2008.

[22] D. Cox and S. John, "A statistical method for global optimization," in *Systems, Man and Cybernetics, IEEE International Conference on*, Oct 1992, pp. 1241–1246 vol.2.

[23] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, 2010, pp. 1015–1022.

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