Variance Modulated Task Prioritization in Whole-Body Control

Ryan Lober, Vincent Padois, Olivier Sigaud

Variance Modulated Task Prioritization in Whole-Body Control

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

***Abstract*— Whole-Body Control methods offer the potential to execute several tasks on highly redundant robots, such as humanoids. Unfortunately, task combinations often result in incompatibilities which generate undesirable behaviors. Prioritization techniques can prevent tasks from perturbing one another but often to the detriment of the lower precedence tasks. For many tasks, static prioritization is not necessary or even appropriate because tasks can often be achieved in variable ways, as in reaching. In this paper, we show that such task variability can be used to modulate task priorities during execution, to temporarily deviate certain tasks as needed, in the presence of incompatibilities. We first present a method for mapping from task variance to task priority and then provide an approach for computing task variance. Through three common conflict scenarios, we demonstrate that mapping from task variance to priorities reactively solves a number of task incompatibilities.**

I. INTRODUCTION

Highly redundant robots, such as humanoids or anthropomorphic platforms, provide the capability of executing several tasks simultaneously. Unfortunately, this versatility comes at the cost of control difficulty due to the high dimensionality and undetermined nature of the inverse control problem. Over the past few decades, Whole-Body Control, or WBC, techniques have emerged as effective means of controlling these systems, by allowing multiple tasks to be specified simultaneously and utilizing their full capacity [1], [2], [3], [4].

The execution of multiple tasks can induce unwanted behaviors due to incompatibilities between them. Typically, priorities are used to ensure that safety-critical tasks, such as balancing, remain unperturbed by incompatibilities with uncritical tasks like reaching. These priorities may be strict [4] or soft [3] hierarchies.

While distinctions between tasks deemed safety-critical or not are made with relative ease, discriminating between uncritical tasks is less trivial. In many situations it is impossible to analytically justify the prioritization of one task over another and as a result, tasks priorities are commonly subject to arbitrary manual tuning. In many cases, such static prioritization is too restrictive and can engender additional incompatibilities that could be otherwise avoided. For anthropomorphic or humanoid robots, generating tasks for the end-effectors (EE) is crucial for manipulation and interaction with the robot's environment; however, manipulation tasks are generally not considered safety-critical and are consequently difficult to prioritize.

EE tasks generally require that the robot pass through one or more waypoints, as in goal reaching. These tasks possess the property of *task redundancy* [5], which implies that there exist infinitely many ways of passing from one waypoint to the next. In practice, trajectories passing through the waypoints are generated and fed to the controller. However, these trajectories do not need to be followed with the same precision as near the waypoints. Works from the field of imitation learning have approached incompatibility resolution for single EE tasks, i.e. external perturbations and poorly formed reference trajectories, by exploiting the demonstrated task's redundancy to regulate the task controller's impedance gains [5]. These studies determine task redundancy from the variance of the movement demonstrations. The task controller is typically some version of the Dynamical Movement Primitive, or DMP, [6] and contains an attractive Proportional-Derivative (PD) term along with a learned forcing term, for example,

$$\boldsymbol{\xi}_{des}(t+\delta t) = K_p\boldsymbol{\epsilon}(t) + K_d\dot{\boldsymbol{\epsilon}}(t) + f. \tag{1}$$

Here $\boldsymbol{\xi}_{des}(t+\delta t)$ is the desired task-space acceleration term, $\boldsymbol{\epsilon}(t)$ and $\dot{\boldsymbol{\epsilon}}(t)$ are the current pose error and its derivative, K_p and K_d, their proportional and derivative gains respectively, and f, the forcing term learned via regression techniques from multiple movement demonstrations [6]. Task variance is measured from the variability of the learned motions and may be adapted based on new demonstrations [7], [8], [9]. An inverse relationship between the task variance and the K_p gain is then formed to regulate the attractor term during the movement. Consequently, when variance is high, the robot is compliant, and when variance is low, the robot is stiff. Variable compliance (a.k.a. gain scheduling), allows the robot to adapt to uncertainties/incompatibilities in its environment. Unfortunately, these conflicts must often be directly integrated into the task controller, making variable compliance, task specific [5]. In addition, the K_p gains may vary by orders of magnitude for a single task [10]. Nevertheless, there is clearly some relationship between the variance of a task and its execution [11].

In this paper, we employ task variance to modulate soft task hierarchies, represented by continuous real valued weights, within a Whole-Body (WB) controller, rendering it more robust to incompatibilities, perturbations and poorly designed reference trajectories. By varying the task weights during execution, the WB controller can temporarily deviate high variance tasks in the presence of incompatibilities on an as-needed basis. This allows the robot to reactively solve a

[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

range of task incompatibilities. We demonstrate how variance can be mapped to task weights for the individual Degrees of Freedom (DoF) of a task and develop a method for computing variance for a task if none is available, as is the case with typical trajectory planners. Finally, we test our variable weighting method in three common incompatibility scenarios on a humanoid robot in simulation.

II. METHODS

In this section, we first give a broad overview of WB task-based hierarchical control. We then propose a method of mapping from a task's variance to its weight in order to modulate its priority over the course of execution. A technique for computing variance for a single task is also developed.

A. Whole-Body Control

WB controllers seek to reactively calculate the joint torques, $\boldsymbol{\tau}$, necessary to minimize a combination of task errors using all of the DoF of the given robot. Task errors can be formulated as the difference between task-space reference commands and their joint-space representations,

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

Here $T_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\xi}_i^*, \mathbb{X})$ is an acceleration task error, J_i and $\dot{J}_i$, the task Jacobian and its derivative, $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$, the joint-space variable states and $\boldsymbol{\xi}_i^*$ the reference task-space acceleration to affect for some frame attached to the robot. The dynamic variable, $\mathbb{X} = [\, \ddot{\boldsymbol{q}}^T, \boldsymbol{w}_e^T, \boldsymbol{\tau}^T \,]^T$, groups the joint-space accelerations and torques with the external wrenches, $\boldsymbol{w}_e$. The variable $\boldsymbol{\xi}_i^*$ is commonly provided by a task-level feedforward Proportional-Derivative (PD) controller,

$$\boldsymbol{\xi}_i^*(t + \delta t) = \boldsymbol{\xi}_{des_i}(t + \delta t) + K_p \boldsymbol{\epsilon}_i(t) + K_d \dot{\boldsymbol{\epsilon}}_i(t) \ , \quad (3)$$

where $\boldsymbol{\xi}_{des_i}(t + \delta t)$ is the feedforward frame acceleration term. An optimization problem can then be designed to find the minimum of the weighted sum of n_T task errors, subject to the problem constraints,

$$\begin{aligned} \underset{\mathbb{X}}{\text{argmin}} \quad & \tfrac{1}{2} \textstyle\sum_{i=1}^{n_T} w_i T_i + w_0 T_0 \\ \text{subject to:} \quad & G\mathbb{X} \preceq \boldsymbol{h} \\ & A\mathbb{X} = \boldsymbol{b}. \end{aligned} \quad (4)$$

The dynamic variable, allows the dynamic equations of motion to be represented as the equality constraint, $A\mathbb{X} = \boldsymbol{b}$. Inequality constraints such as $G\mathbb{X} \preceq \boldsymbol{h}$ can account for considerations such as contacts, joint limits, and actuator limits. The importance of each task is governed by its weight w_i, and a regularization task, T_0, is used to ensure a unique optimization solution with $w_0 \ll w_i$. Varying the task weights, and consequently their priorities, generates joint torque commands which favor the minimization of task errors with higher associated weights[1]. Equation (4) can be minimized efficiently using a Linear Quadratic Program. More details on WBC can be found in [12], [2], [3].

[1]It is also common to resolve this optimization hierarchically in order of task priority, projecting the lower priority tasks into the null space of the higher priority tasks. [4].

B. Task Formalism

Here we look at Cartesian goal reaching tasks, and without loss of generality, only their translation components are considered.

Each task follows some trajectory, Υ, which passes through one or more waypoints. A trajectory has two components, its path which consists of a series of vectors of spatial coordinates, $\boldsymbol{r}_i = [x, y, z]$ with $\{i \in \mathbb{N} | 1 \leq i \leq N_r\}$, where N_r is the total number of spatial coordinate vectors, and its temporal evolution, $\boldsymbol{t}$, which dictates the dynamics of the movement.

Looking at these tasks in a general probabilistic fashion, we can use the position vectors $\boldsymbol{r}_i$ as the mean, $\boldsymbol{\mu}_i$, of our task trajectory. The variance of the movement at each timestep, $\boldsymbol{\sigma}_\Upsilon^2(t) = [\sigma_{\Upsilon_x}^2(t), \sigma_{\Upsilon_y}^2(t), \sigma_{\Upsilon_z}^2(t)]$, can be obtained through multiple demonstrations[2] as in [5], [7], [8], [9], or computed from scratch. The concatenation of these position means and variances, respectively yields M_Υ and V_Υ for the given trajectory, Υ.

C. Mapping Variances to Weights

Given a trajectory, Υ, with some variance, V_Υ, we can create a relationship between V_Υ and the task's weight, w_i, at each timestep, t, making the task weights now time and variance dependent within the WB controller. We would like to restrict our variable weight evolution to the $[0.0, 1.0]$ range, therefore all tasks in the WB controller are defined with a baseline weight of 1.0 and we rescale the trajectory variance such that $\{\overline{V}_\Upsilon \in \mathbb{R} | 0 \leq \overline{V}_\Upsilon \leq 1\}$,

$$\overline{V}_\Upsilon = \frac{V_\Upsilon}{\max(V_\Upsilon)}. \quad (5)$$

Equation (5) also ensures that the DoF variances are scaled relative to one another. The variance of each DoF may not be the same, so we map a variance to a weight for each. Therefore, $w_i(\boldsymbol{\sigma}_\Upsilon^2(t))$ becomes the diagonal weight matrix, $W_i(\boldsymbol{\sigma}_\Upsilon^2(t))$, and using a maximum weight factor, β, we can map from variance to weights using this basic approach,

$$W_i(\boldsymbol{\sigma}_\Upsilon^2(t)) = \begin{bmatrix} \frac{1-\sigma_{\Upsilon_x}^2(t)}{\beta} & 0 & 0 \\ 0 & \frac{1-\sigma_{\Upsilon_y}^2(t)}{\beta} & 0 \\ 0 & 0 & \frac{1-\sigma_{\Upsilon_z}^2(t)}{\beta} \end{bmatrix}. \quad (6)$$

Where the variance of the movement is high, $\overline{V}_\Upsilon$ is close to 1 and so the weight/importance of the task diminishes. When the variance is small, $\overline{V}_\Upsilon$ approaches 0 and the importance of the task is at a maximum. The factor β allows us to scale the overall importance of the task relative to the other tasks, while still maintaining variability. For instance, assuming all tasks have a baseline weight of 1, $\beta < 1$ means the variable weight task is less important than the other tasks, while $\beta > 1$ the inverse. This is useful when combining uncritical tasks with safety-critical tasks such as balancing; however, it does not guarantee that the safety-critical task will go unperturbed.

[2]In these works, the covariances of the forcing term basis functions are used.

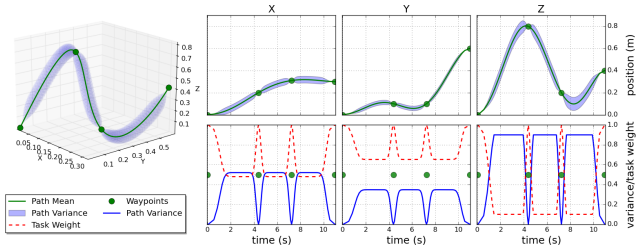


Fig. 1: An example of a 3D task trajectory with variance. This figure shows how variance can be computed given a trajectory, then mapped to the weights of the individual DoF of the task.

In practice, if the variance is too close to 1.0, the weight of the task becomes infinitesimal and the controller no longer executes it. In order to avoid such behavior, the maximum $\overline{V}_\Upsilon$ can be bounded at a value just less than 1.0 (e.g., 0.99 is used in this study).

D. Computing Variance

Historically, task variance has been calculated from multiple demonstrations of the same movement [5], [7]. Unfortunately, demonstration data is not always available, and it is advantageous to be able to compute task variance when we only have one example, as is the case with trajectory generators. Here, we use a covariance function for this purpose.

Covariance functions are commonly used in the field of Gaussian Process Regression (GPR) [13] and allow one to calculate the variance of a new output point in an existing data set by paving the input data space with kernel functions. Here we use Gaussian kernels[3]:

$$k_i(m) = \sigma_k^2 \exp\left(\frac{-(m-c_i)^2}{2l_k^2}\right). \tag{7}$$

The variable σ_k^2 is the maximum allowable covariance, l_k is the length parameter which influences how much adjacent kernel centers, c_i, influence each other and m is the input value for which we wish to calculate the kernel output. Typically one kernel is centered on each input datum.

Given some new input, m^*, and N_k total kernels, we can calculate the variance of its output as [13],

$$\text{var}(m^*) = K_{**} - K_* K^{-1} K_*^T \tag{8}$$

where,

$$K = \begin{bmatrix} k_1(c_1) & k_2(c_1) & \cdots & k_{N_k}(c_1) \\ k_1(c_2) & k_2(c_2) & \cdots & k_{N_k}(c_2) \\ \vdots & \vdots & \ddots & \vdots \\ k_1(c_{N_\lambda}) & k_2(c_{N_k}) & \cdots & k_{N_k}(c_{N_k}) \end{bmatrix}, \tag{9}$$

$$K_* = \begin{bmatrix} k_1(m^*) & k_2(m^*) & \cdots & k_{N_k}(m^*) \end{bmatrix} \tag{10}$$

and

$$K_{**} = k_{m^*}(m^*)\ , \tag{11}$$

[3]Also referred to as the squared exponential in GPR literature.

with $k_{m^*}(m^*)$, a kernel centered and evaluated on the new input, m^*. In this formulation, the kernel centers are points of zero variance, and the variance of the intermediate points is calculated by evaluating (8) between the kernel centers. In terms of goal reaching tasks, variance should be zero at the waypoints meaning that kernel centers should be placed on each one.

Given a single demonstrated trajectory, we can only confidently interpret two waypoints, one at the beginning of the movement and one at the end, based on the assumptions that the trajectory was generated from the EE starting state, and that the final state of the trajectory represents the goal of the movement. If more waypoints are given, such as in the case of programmed trajectories, then they too may be used. The ensemble of waypoints, $\boldsymbol{\lambda}_j$, can be indexed by the order in which they are to be attained, $\{j \in \mathbb{N} | 1 \leq j \leq N_\lambda\}$, where N_λ is the total number of waypoints.

We define our kernel centers on the indexes j of the waypoints inferred from the trajectory. We can then create $\boldsymbol{m}$, our evaluation domain, by resampling the position vectors $\boldsymbol{r}_i$ as $\boldsymbol{r}_m$ such that, $\{m \in \mathbb{R} | 1 \leq m \leq N_\lambda\}$. Now, to calculate the variance of some position $\boldsymbol{r}_{m^*}$, (8) is evaluated at m^*, the resampled index of $\boldsymbol{r}_{m^*}$.

For each DoF of the movement, $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ we must calculate the kernel parameters, σ_k^2 and l_k. The variance of the position values for each DoF can be used to calculate their individual maximum allowable variances, $\boldsymbol{\sigma}_k^2 = [\sigma_{k_x}^2, \sigma_{k_y}^2, \sigma_{k_z}^2]^T$ using,

$$\boldsymbol{\sigma}_k^2 = \frac{\sum_{i=1}^{N_r} (\boldsymbol{r}_i - \overline{M}_\Upsilon)^2}{N_r - 1}. \tag{12}$$

Again, N_r is the total number of positions vectors, $\boldsymbol{r}$, and $\overline{M}_\Upsilon$ is the mean of each DoF of the movement. The length parameter, l_k, can be set using,

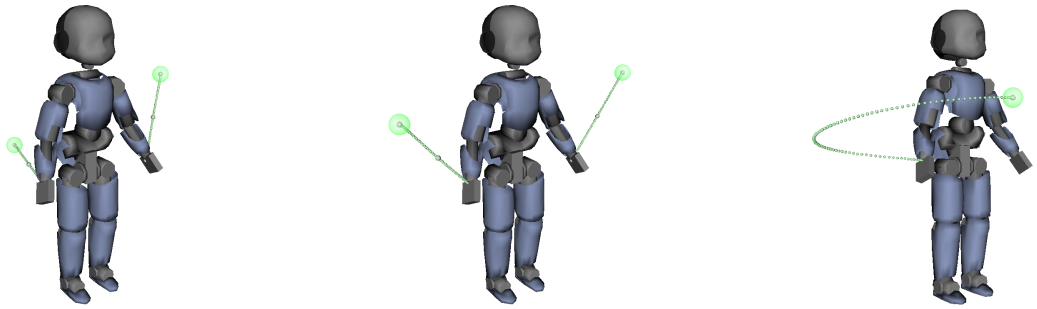
$$l_k = \frac{N_\lambda}{\alpha_l}\ , \tag{13}$$

where α_l is some scaling coefficient; here we use $\alpha_l = 10.0$. Figure 1 shows a 3D Cartesian trajectory with 4 waypoints and the variance computed using the aforementioned techniques. Given this variance, we can map to task weights using (6), for each DoF. This is shown by the DoF plots in Fig. 1.

III. EXPERIMENTAL SETUP

To test the efficacy of using variable task weights in a WB controller, three simulated scenarios are presented to highlight some common issues encountered when combining multiple incompatible tasks. In each scenario, a set of tasks is hand-coded for a simulation of the humanoid robot, iCub, which possesses 32 actuated DoF[4]. The XDE physics simulator and environment [14], [15] is used in this study. Successful task combination is characterized as the proximity of the hand task frames to their respective goal locations within a margin of 3.0 cm. This margin is

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



(a) Constrained Configuration (b) Workspace Violation (c) Balance Perturbation

Fig. 2: Three common multi-task incompatibility scenarios. The desired hand task trajectories are indicated by the green markers. Medium size spheres represent waypoints, and large transparent spheres represent the final waypoints or goals.

represented by the large transparent spheres at the end of the hand trajectories in Fig. 2, and has been selected solely to aid in visualization. More precise margins can be applied without loss of generality. We present the execution of these scenarios using both static and variable task weights. The task variances are computed using (8) with a maximum allowable scaled variance of 0.99.

A. Constrained Configuration

In this first scenario, Fig. 2(a), three principle tasks are combined to force the robot into a constrained configuration. Such configurations commonly occur on highly redundant systems when multiple tasks require the same DoF. In humanoids, this often occurs due to solicitation of the torso DoF. The first standing task maintains the center of the robot's waist at a constant height with a static weight of 1.0. The two variable weight tasks are associated with the left and right hands, specifically the center of the base of the palms. These tasks are defined by trajectories passing through waypoints at the beginning, middle and end of the movements. The task objectives are for them to attain the final waypoints, or goal positions, while passing through the other waypoints. The left and right hand tasks last 6.4s and 6.1s respectively, and are executed with $\beta = 1.0$.

B. Workspace Violation

The second scenario, Fig. 2(b), combines the same three primary tasks as in Sec. III-A; however, this time the hand trajectory goal positions are further apart than the maximum workspace of the robot (in this standing configuration). This scenario is designed to represent a typical workspace conflict during picking procedures. The trajectories pass through waypoints at the beginning, middle and end of the movements. The left and right hand tasks last 6.3s and 6.2s respectively, and are executed with $\beta = 1.0$. When one of the hands attains its goal position, that is, within 3.0 cm of the final waypoint, that task is deactivated (i.e. the object has been picked). Task deactivation means that it no longer contributes to the control solution, or equivalently, that its weight is set to 0.0.

C. Balance Perturbation

Here we combine Zero Moment Point (ZMP) balancing [16] with a right hand task (see Fig. 2(c)). The objective of the ZMP balancing task is to maintain the Center of Pressure, or CoP, (x, y) coordinates at $(0, 0)$, its initial position. The right hand follows a sweeping trajectory from the hand's starting waypoint to its end waypoint - no intermediary waypoints are considered. This scenario is meant to replicate activities similar to wiping surfaces. The right hand task lasts 8.6s and is executed with $\beta = 10.0$.

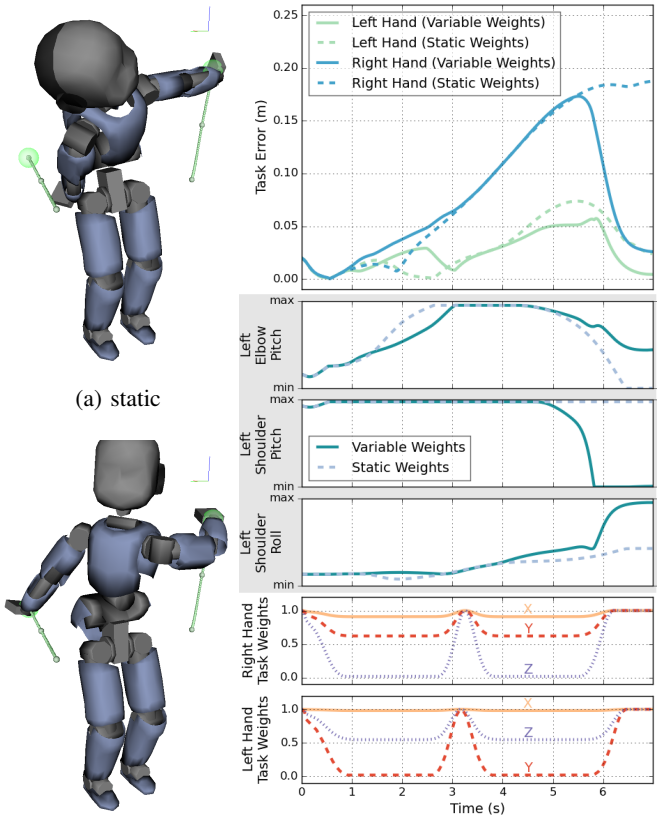
IV. RESULTS

In this section we provide the results of the scenario simulations described in Sec. III. A video presenting these experiments and their results can be found in the attachments of this submission.

A. Constrained Configuration

When the two hand tasks are combined with static weights, we can see in Fig. 3(a) that the left hand task achieves its goal location, contrary to the right hand task. This occurs because individually, the hand tasks require the torso to rotate left and right; therefore, when they are combined this DoF is constrained between the two. The arm DoF attempt to compensate for this reduction in redundancy by moving to their limits, and forcing the robot into a constrained configuration. This is shown in the left arm DoF plots in Fig. 3(c). Consequently, the right hand task is no longer feasible and incurs high task errors at both the middle and goal waypoints due to its combination with the left hand task. This can be observed in the task error plot of Fig. 3(c). The waypoints along the trajectory are indicated by the peaks in the task weight curves in Fig. 3(c).

In Fig. 3(b) the robot has successfully accomplished its tasks through the use of variable weights. By looking at the left arm DoF plots in Fig. 3(c) we can see that the right hand task weight increases approximately 0.25s prior to the left hand weight, forcing the robot to dedicate more DoF to its execution and causing the left arm elbow pitch, shoulder pitch and shoulder roll to deviate. These deviations pull the left arm DoF away from their limit values, freeing these

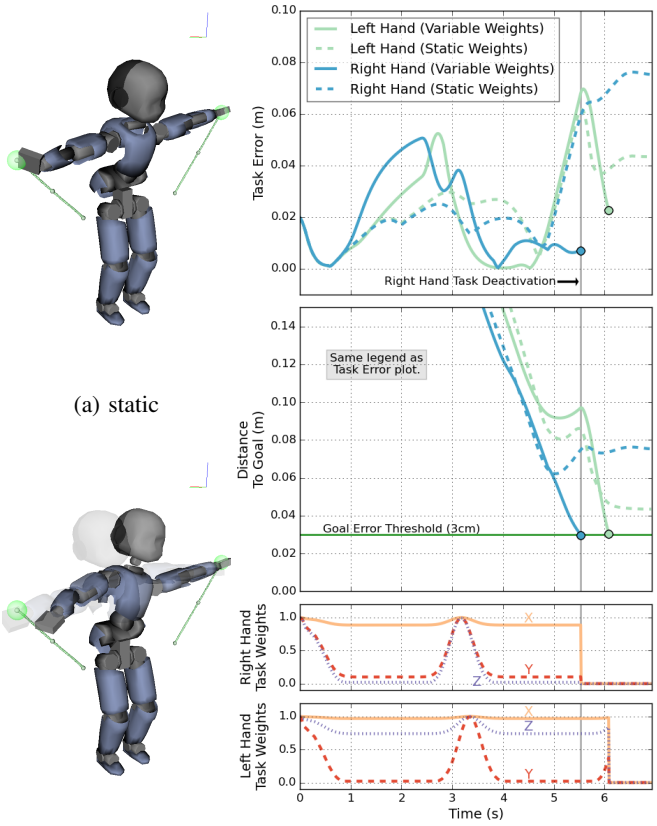


(a) static

(b) variable

(c)

Fig. 3: The constrained configuration scenario. Figures (a) and (b) show the task combination results using static and variable weights respectively. The plots in (c) provide the simultaneous evolution of various task parameters.



(a) static

(b) variable

(c)

Fig. 4: The workspace violation scenario. See Fig. 3 description for layout details.

articulations for the left hand movement when its weight increases.

B. Workspace Violation

Figure 4(a) shows the static execution of the two hand tasks and although the hands seem to reach their goal positions, close inspection of the distance to goal plot in 4(c) shows that they never attain the 3.0cm error threshold limit. As a result, they rest in a local minimum between their two objectives.

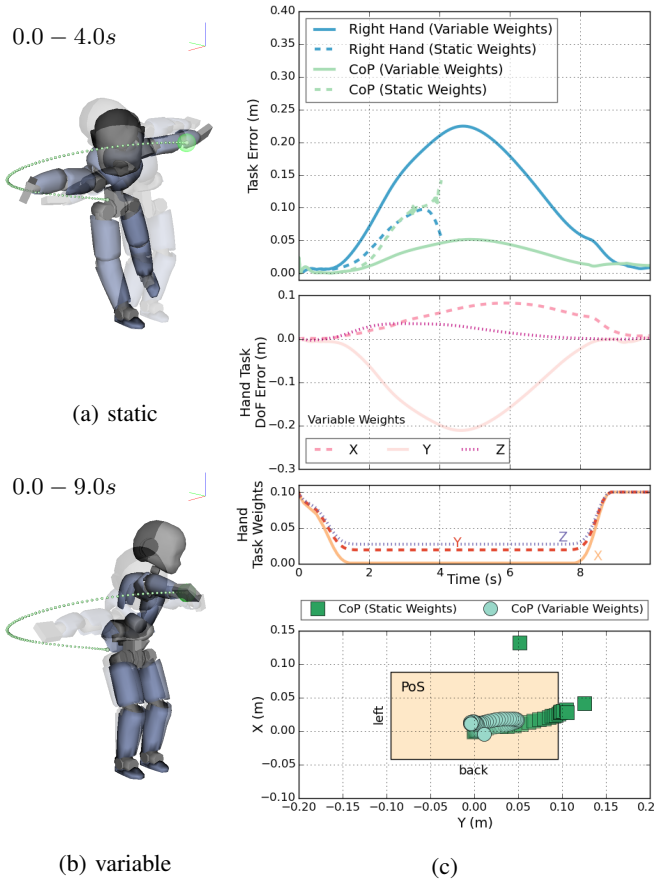
When variable weights are applied to the simultaneous execution of the two hand tasks, the robot achieves its right hand goal first, thereby deactivating the right hand task, and then proceeds to finish the left hand task; this is shown in Fig. 4(b). The instants that the hand tasks are deactivated can be seen in the task error and distance to goal plots, and are indicated by circular markers.

In both the right and left hand movements, the y directional component develops large errors near the goal locations. The errors are roughly equivalent (see Fig. 4(c) static task error plot) and therefore whichever task has the largest w_{y_i} dominates in the WB controller output - the right hand task in this case (see Fig. 4(c) hand task weight plots). Once the right hand task is deactivated, all conflicts are removed and the left hand task is able to recuperate its accumulated error and be deactivated as well.

C. Balance Perturbation

Figures 5(a) and 5(b), show the balance perturbation results. Using static weights for the right hand task results in a loss of balance and ultimately a failure for both tasks; this can be seen in Fig. 5(a). We can confirm this loss of balance by observing that the CoP moves outside of the Polygon of Support, or PoS, in Fig. 5(c). Despite the ZMP balancing being 10x more important than the right hand task, it still fails because the accumulated error at the apex of the sweeping movement generates large enough accelerations in the y direction to perturb the ZMP balancing.

In the variable weight case, we can see in Fig. 5(b) that the robot successfully attains the goal position of the hand task while remaining balanced. The task error plot shows that, the right hand task incurs a large amount of error as in the static case, but because this occurs during a period of high variance, this error only partially perturbs the ZMP balancing. The CoP is deviated somewhat from its goal location in order to compensate for some of the right hand error but it remains safely within the PoS as shown in Fig. 5(c).



(a) static

(b) variable

(c)

Fig. 5: The balance perturbation scenario. See Fig. 3 description for layout details.

V. CONCLUSION

Regulating task weights based on their variance is a powerful concept, which when coupled with WBC methods, can solve difficult control problems on-line. The use of variable weights diminishes the need for manual tuning of task priorities, and provides WB behaviors which are more robust to incompatibilities, perturbations and poorly designed reference trajectories.

In this paper, we presented a simple technique for utilizing task variance as a means of modulating task weights automatically in a WB controller. These variable weights permit the WB controller to temporarily deviate high variance tasks in the presence of incompatibilities. Through three emblematic scenarios, we showed how variable task weights resolve a broad set of issues encountered in multi-task execution with minimal tuning and in a reactive manner. In addition to the variance to weights mapping, we developed a method of computing variance for a single trajectory demonstration using a covariance function (8). This tool is essential in cases where only one trajectory has been provided for the task, as in trajectory generation.

High task variance allows one to handle conflicts between tasks but provides no guarantee that the tasks will be accomplished. If an incompatibility occurs when all tasks require low variance, or high priority, then our method will not work and some form of planning must occur. In [17], we show that by optimizing tasks over their entire execution, we can ensure task completion; however, this method is time consuming. In the future, we will investigate how to combine such global optimization methods with variance modulated weighting, to provide a fast and robust task control framework which can assure task realization.

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] M. de Lasa and A. Hertzmann, "Prioritized optimization for task-space control," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, no. 2, pp. 5755–5762, Oct 2009.

[2] L. Saab, N. Mansard, F. Keith, J.-Y. Fourquet, and P. Souères, "Generation of dynamic motion for anthropomorphic system under prioritized equality and inequality constraints," in *IEEE International Conference on Robotics and Automation*, Shangai, China, May 2011.

[3] 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*, 2011.

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

[5] S. Calinon, I. Sardellitti, and D. G. Caldwell, "Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies," in *IEEE International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 249–254.

[6] 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.

[7] P. Kormushev, S. Calinon, and D. G. Caldwell, "Approaches for learning human-like motor skills which require variable stiffness during execution," in *IEEE International Conference on Humanoid Robots*, 2010.

[8] F. Stulp, J. Buchli, E. Theodorou, and S. Schaal, "Reinforcement learning of full-body humanoid motor skills," *IEEE-RAS International Conference on Humanoid Robots*, pp. 405–410, Dec 2010.

[9] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, April 2011.

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

[11] E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature neuroscience*, vol. 5, no. 11, pp. 1226–35, Nov 2002.

[12] 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*, 2009.

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

[14] 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," in *Joint International Conference on Multibody System Dynamics*, 2012.

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

[16] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2003, pp. 1620–1626.

[17] R. Lober, V. Padois, and O. Sigaud, "Multiple task optimization using dynamical movement primitives for whole-body reactive control," in *IEEE International Conference on Humanoid Robots*, 2014, pp. 1–6.