Bio-inspired balance controller for a humanoid robot*

François Heremans[1], Nicolas Van der Noot[1,2], Auke J. Ijspeert[2] and Renaud Ronsse[1]

Abstract— **Humanoid robots are gaining much interest nowadays. This is partly motivated by the ability of such robots to replace humans in dangerous environments being specifically designed for humans, such as man-made or natural disaster scenarios. However, existing robots are far from reaching human skills regarding the robustness to external perturbations required for such tasks, although torque-controlled and even bio-inspired robots hold new promises for research. A humanoid robot robustly interacting with its environment should be capable of handling highly uncertain ground structures, collisions, and other external perturbations. In this paper, a 3D bio-inspired balance controller is developed using a virtual lower limbs musculoskeletal model. An inverse muscular model that transforms the desired torque patterns into muscular stimulations closes the gap between traditional and bio-inspired controllers. The main contribution consists in developing a neural controller that computes the muscular stimulations driving this musculoskeletal model. This neural controller exploits the inverse model output to progressively learn the appropriate muscular stimulations for rejecting disturbances, without relying on the inverse model anymore. Two concurrent approaches are implemented to perform this autonomous learning: a cerebellar model and a support vector regression algorithm. The developed methods are tested in the Robotran simulation environment with COMAN, a compliant child-sized humanoid robot. Results illustrate that – at the end of the learning phase – the robot manages to reject perturbations by performing a full-body compensation requiring neither to solve an inverse dynamic model nor to get force measurement. Muscular stimulations are directly generated based on the previously learned perturbations.**

I. INTRODUCTION

Humanoid robotics has gained much interest during the last decades. This interest for human-like robots is partly driven by a strong incentive: evolving in environments made for humans is challenging for a robot due to the numerous artifacts that were specifically designed for us (stairs, doors, levers, etc.). The quest for humanoid robots that can move and act in those complex environments is therefore partly motivated by scenarios where these robots could replace humans. This is particularly relevant in hazardous environments such as man-made or natural disaster scenarios.

However, the world outside the lab is an uncontrolled environment for legged humanoid robots, with unexpected ground levels, potential collisions and other external perturbations. Coping with these unexpected perturbations requires a robust balance controller. Despite major advances in the field of robotics and automation, robots are still far from reaching the superb ability of humans to balance in challenging situations. Taking inspiration from the human neuromechanical apparatus thus represents a promising research avenue to bridge this gap. This idea has already been applied for dynamic walking in simulation. For instance, a lower-limbs musculoskeletal model comprising virtual muscles was developed in [1]. Still in simulation, this model was further incremented in [2] by the addition of a bio-inspired oscillatory neural controller modulating the forward speed. Nevertheless, the simulation/reality gap represents a challenge for traditional controllers. Bio-inspired controllers can help in reducing this gap by providing natural compliance to cope with the world non-idealities. For instance, [3] shows an almost straightforward transfer of the controller in simulation to the real humanoid robot.

Alternatively, legged robots also provide a fantastic tool to improve our understanding of the human neuromusculoskeletal apparatus. Indeed, robots offer to emulate isolated components of this apparatus and so to test hypotheses about their behavior. Moreover, robots can serve to simulate pathologies and thus investigate their consequences. Finally, bipedal robots offer the opportunity to generate a large amount of data and thus to study the system sensitivity to a large set of neuromechanical parameters.

Regarding postural control, different balancing strategies were identified from human experiments. In the case of small to medium perturbations, humans can maintain balance using body coordination only, so that no stepping is required [4]. However, the neuro-motor pathways leading to such postural responses are not straightforward. Over the past years, simplified models were elaborated to capture this complexity. For instance, [5] introduced the Cerebellum Model for Articulation Control (CMAC), a simplified cerebellum model for robot control intended to provide motor learning capabilities.

In addition, the recent developments of torque controlled robots equipped with compliant actuators, i.e. series elastic actuators, provide new tools to validate the aforementioned neurological models. These new platforms also enhance the use of compliant controllers, allowing robust interactions with the environment. For example, [6] uses a passivity-based admittance (compliant) controller for stabilization, making the robot compliant to external steady-state perturbations. Similarly, [7] modulates ground applied forces for balancing. Such compliant controllers are well adapted to

*This work was supported by the Belgian F.R.S.-FNRS (Aspirant #6809010 awarded to FH, Aspirant #16744574 awarded to NVdN, Crédit aux Chercheurs #6809010 awarded to RR) and by the European Community's Seventh Framework Programme under Grant 611832 (WALK-MAN)

[1] F. Heremans, N. Van der Noot and R. Ronsse are with the Center for Research in Mechatronics, Institute of Mechanics, Materials and Civil Engineering, Université catholique de Louvain, Belgium.

[2] N. Van der Noot and A. J. Ijspeert are with The Biorobotics Laboratory, School of Engineering, Institute of Bioengineering, Ecole Polytechnique Féderale de Lausanne, Switzerland

situations with dynamic interactions such as balancing or physical contacts with humans.

In this paper, we focus on perturbations that do not require stepping to recover equilibrium. We merge bio-inspiration and classical control techniques to achieve a novel controller. In a nutshell, the contributions of this paper are: (i) the implementation of an inverse muscular model, (ii) the development of a neural controller using machine learning techniques to produce a regression model of the muscular stimulations and (iii) the development of a training chain based on an impedance controller. To the best of our knowledge, this contribution is the first to report the use of machine learning techniques to drive a 3D musculoskeletal model of the legs for balance control. The developed method is validated in simulation. This paper is structured as follows. In Section II, the control framework is outlined. The neuromusculoskeletal model and the learning mechanisms are described. Section III details the simulation protocol used to validate the proposed controller. Section IV describes the simulation results. Finally, the simulation outcomes are discussed in Section V.

II. POSTURAL CONTROL FRAMEWORK

This section details the framework providing a full-body compensatory motion to counter 3D perturbations. A neuromusculoskeletal chain inspired by the human neuromechanical apparatus is introduced to capture the robot controller (top part of Fig. 1). This bio-inspired chain is composed of two modules. A neural controller implemented by a regression engine (machine learning) is in charge of generating virtual muscular simulations. Using these stimulations, a 3D musculoskeletal model generates joint torques applied to the lower limbs. An impedance controller and an inverse muscular model (lower part of Fig. 1) were further implemented in order to generate the reference stimulations being required for the learning process of the neural controller. The impedance controller also drives upper body movements for which no musculoskeletal model was developed.

A. Musculoskeletal model

A virtual musculoskeletal system was developed to drive the lower-limbs degrees of freedom (Fig. 2), as an extension of the musculoskeletal model developed by [1], [8].

The fundamental building block of the muscular model depicted in Fig. 2 is the Hill-type muscle model (Fig. 2(b) and [9]). Each muscle tendon unit (MTU) consists of two main elements: an active contractile one (CE) and a passive series elastic one (SE). On top of that, a parallel-elastic element (PE) and a buffer elasticity element (BE) prevent the muscle from collapsing on itself or overstretching when it leaves its nominal operation range.

The sagittal muscles – soleus (SOL), tibialis anterior (TA), gastrocnemius (GAS), vastus (VAS), hamstring (HAM), gluteus (GLU) and hip flexor (HFL) – were adapted from [1] and the hip adduction (HAB)/abduction (HAD) muscles from [8]. The model was further incremented with four extra mono-articular muscles (HER, HIR, EVE, INV) for the hip external/internal rotations and the foot eversion/inversion,

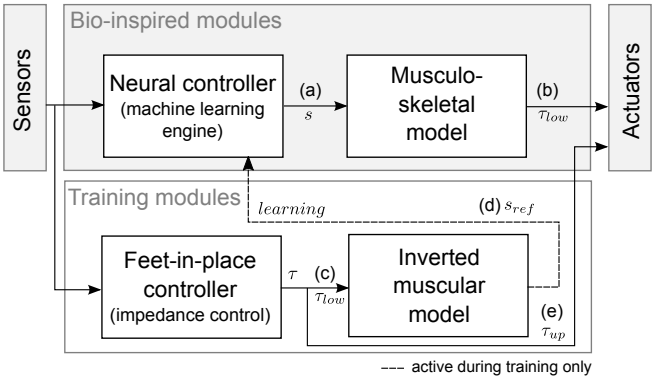

Fig. 1. Architecture of the main controller. Based on the sensed perturbation, the neural controller generates muscular stimulations (a) to drive a musculoskeletal model. The resulting joint torques (b) actuate the robot lower limbs. In parallel, an impedance controller computes desired torques (c) being transformed into virtual stimulations (d) by an inverse muscular model. These are used by the neural controller during the learning phase. The impedance controller further computes the upper-limb reaction torques (e), for which no muscular model was developed.

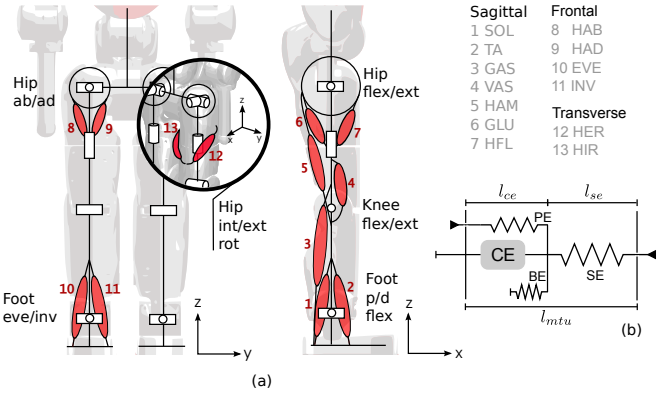

Fig. 2. (a) 3D lower limbs musculoskeletal model comprising 13 Hill-type muscles (b) per leg.

respectively. Similarly to [8], the actuation scheme is simplified by assuming the full decoupling of the sagittal, frontal and transverse planes. This assumption does not capture the whole complexity of the human motor system but can be relaxed later with no major consequences for our framework. Table I provides the anatomical parameters of the four extra muscles using the template provided in [1] (see [8] for the other muscles). These muscle parameters were estimated using the lower-limbs model from [10] in OpenSim [11]. Dynamic scaling was used to tune the muscular parameters to the particular robot being used [12][13] (see Section III for the robot description).

Each virtual muscle is simulated by a direct model that transforms muscle stimulations into muscle forces further transformed into joint torques as a function of their insertion points. At each time step, two operations are repeated: (i) updating each muscle state based on the local skeletal configuration and the muscular stimulations; and (ii) computing the resulting joint torques based on the muscle forces and skeletal configuration. More precisely, a muscular iteration

TABLE I

HIP EXTERNAL/INTERNAL ROTATION AND FOOT EVERSION/INVERSION MUSCLES PARAMETERS

Muscle	HER	HIR	EVE	INV
F_{max} $[N]$	177	283.2	247.8	318.6
v_{max} $[l_{opt}/s]$	18.36	18.36	18.36	18.36
l_{opt} $[cm]$	1.708	3.416	2.135	2.135
l_{slack} $[cm]$	2.135	2.989	10.675	12.81
r_0 $[cm]$	1.708	1.281	1.281	0.854
ϕ_{max} $[deg]$	-	-	-10	5
ϕ_{ref} $[deg]$	10	-20	-5	-10
ρ $[-]$	1	0.7	0.7	0.7

goes through the following steps (see Fig. 2(b) and [1] for details):

1) Updating the muscle unit length l_{mtu} based on the current joint state.
2) Updating the muscle internal forces: the parallel F_{pe} and buffer elastic F_{be} forces based on the length l_{ce} computed during the previous iteration.
3) Updating the force-length and force-velocity relationships f_l and f_v.
4) Updating the muscle contraction speed v_{ce}.
5) Time integrating v_{ce} to obtain the contractile element length l_{ce}.

For each leg, the transformation of muscle forces into joint torques can be written as follows:

$$\tau = R \cdot F \tag{1}$$

with $\tau \in \mathcal{R}^{6\times1}$ the joint torques (each leg has 6 degrees of freedom), $F \in \mathcal{R}^{13\times1}$ the muscular forces (each leg has 13 muscles) and $R \in \mathcal{R}^{6\times13}$ the moment arm matrix depending on model parameters (pennation angles, etc.) and on the current skeletal configuration.

B. Neural controller

The neural controller is implemented by a regression engine that generates a muscular stimulation s (Fig. 1 (a)) for each individual muscle. Once trained, this engine is expected to produce adequate stimulations as a function of the sensed perturbations. Two algorithms were implemented: CMAC (Cerebellum Model for Articulation Control), a bio-inspired neural network modeling the cerebellum organization [5]; and SVR (Support Vector Regression), a modern and powerful regression technique [14]. Both implementations share the same learning objective, i.e. solving the regression problem of finding appropriate muscular stimulations corresponding to a given sensory input (Supervised Learning). Supervised learning is used as a first step to assess the feasibility of muscular control during balacing. Moreover, either CMAC or SVR can only generate a 1D output, so that one learning engine should be connected to each muscle. However, both legs being identical, symmetrical muscles can be actuated by the same engine but with different inputs such that only one module per muscle type is required. In order to comply with the body symmetry, sensory inputs in the frontal and transverse plane must be mirrored, e.g. if one sensory input is y_{CoM} (the lateral displacement of the Center of Mass), the right leg engines receive y_{CoM} while the left leg engines receive $-y_{CoM}$. The same applies to the other frontal/transverse signals.

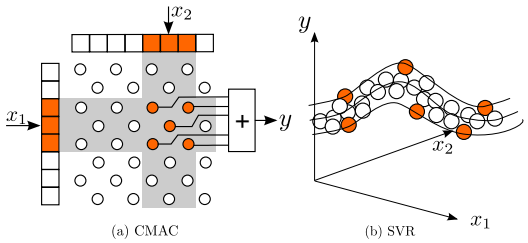

Fig. 3. (a) CMAC model: each input range is discretized. A specific multidimensional input activates a fixed number of input neurons (the orange cells of x1 and x2 in this figure). Based on the activated input neurons, specific core neurons are recruited. Their respective weights are summed to produce the output y. (b) SVR: some data points are elected to become support vectors.

1) Cerebellum model (CMAC): The cerebellum is known to play a major role in motor adaptation [15]. CMAC is a biologically relevant neural network model of the cerebellum providing online learning capabilities. This model captures the reinforcement learning mechanism of the cerebellar Purkinje cells. It builds up a large vector of weights, i.e. a lookup table (LU) with high plasticity. During prediction, depending on the current inputs, some weights are appropriately recruited and summed to produce the output (see Fig. 3(a) for a simplified representation). A thorough description of the gain recruiting and learning mechanisms is available in [15]. In brief, when training is enabled, the core neurons weights evolve according to the following rule:

$$w_k \leftarrow w_k + \alpha\frac{\bar{x} - x}{n_a} \quad (k = 1, \ldots, n_a) \tag{2}$$

with w_k being one of the gains recruited during the previous prediction, α the learning rate, $\bar{x}$, the reference stimulation, x the predicted stimulation and n_a the number of association units (AU), i.e. the number of gains recruited for the prediction.

2) Support Vector Regression (SVR): SVR is an offline learning method. It solves a convex optimization problem so it always converges to the global optimum. The model is generated by selecting appropriate data points as support vectors (see Fig. 3(b)). When training is enabled, the current sensory inputs and the associated reference stimulations (from the training modules, see next section) are added to the learning dataset. This extended dataset is then used to recompute the regression model. The ϵ-SVR algorithm provided by LIBSVM [16] was used in this contribution. ϵ-SVR requires the tuning of four parameters in order to provide an appropriate regression model: p the tolerated precision, C the penalization cost, γ the kernel width and K the kernel type. A grid search cross-validation procedure with 10 folds and a radial basis function (RBF) kernel was carried out to identify the optimal regression parameters γ

(kernel width) and p (tube size). Each sensory input data was scaled to $[-1, 1]$ prior to optimization.

The regression engines, CMAC and SVR, receive the following sensory inputs: (i) all sagittal muscles and the hip int/ext rotation muscles receive the CoM forward position and velocity (x_{CoM}, $\dot{x}_{CoM}$), the CoM lateral displacement (y_{CoM}), and the hip/foot pitch positions. Additionally, the VAS muscle receives the knee position. (ii) Frontal muscles receive y_{CoM}, $\dot{y}_{CoM}$, and the hip/foot roll positions.

C. Reference stimulations

The previously mentioned machine learning algorithms require reference stimulations in order to infer a regression model. These references are computed using two modules (see lower part of Fig. 1): (i) an impedance controller generating reference torques and (ii) an inverse muscular model transforming these torques into muscular stimulations. The error between the reference stimulations (from the inversion module) and the predicted ones (from the neural controller) is continuously monitored. In normal operation, i.e. once the learning phase converged, the neural controller generates adequate stimulations for the muscles such that it is able to autonomously control the lower-limbs. However, if the error goes above a given threshold, the inversion module takes over the control while learning is enabled again for the corresponding muscle. The *cognitive control ratio* is defined as the proportion of simulation time for which the system is under control of the impedance controller and the inversion module (i.e. with learning being active). As the regression model is trained, this ratio is expected to decrease, capturing that the neural controller gradually becomes autonomous.

D. Compliant impedance controller

Coping with perturbations requires highly coordinated body motions. Those motions require accurate torque trajectories for a large set of scenarios, difficult to obtain from human data. Therefore, we implemented the full-body compliant force controller introduced in [7]. This controller was selected because it requires neither an inverse kinematic nor an inverse dynamic model. It can handle an arbitrary number of contact points with the environment and only requires the computation of the direct kinematic model of each contact point [7]. Stability is maintained using virtual feedback forces applied to the Center of Mass (CoM). The robot modulates contact forces with the environment to achieve these virtual forces. A complete description of the algorithm goes beyond the scope of this paper, so that only the main computational steps are outlined below:

1) Computing the forward kinematic model for all contact points with the environment (position and Jacobian with respect to the CoM) and CoM position/velocity.
2) Computing the user force f_u as a proportional-derivative (PD) feedback acting on the CoM.
3) Computing the environment applied forces with gravity compensation as $f_p = f_u + Mg$, with M the robot total mass and g the gravitational acceleration.
4) Computing the position of the desired Center of Pressure (CoP) and solving a linear optimization problem distributing the contact forces.
5) Transforming contact forces into joint torques using the Jacobian matrices.

In the present contribution, the reference torques being obtained after this step are fed to an inverse muscular module that generates the corresponding muscular stimulations.

E. Inverse muscular model

Inverting the direct muscular model outlined in section II-A requires two steps: (i) solving the over-actuation problem for each leg, i.e. transforming the three sagittal torque references (hip, knee and ankle pitch) and three non-sagittal torques (hip yaw, hip and ankle roll) into 13 muscles forces; and (ii) transforming these desired muscle forces into the corresponding muscle stimulations.

Because multiple muscles actuate the same joint, the human musculoskeletal apparatus is redundant. Inverting equation (1) is indeed providing an infinite amount of solutions, because the matrix R has rank 6. Isolating a specific solution thus requires using a particular optimization technique. In addition, the inversion should further obey some constraints. Indeed, a muscle can only pull, i.e. provides positive force, and saturates to a given maximal force. Consequently, each force F_i should be bounded between 0 and F_{max}.

1) Solving over-actuation: The inversion should be computed in real-time since it is intended to work in a real-time learning framework. A quick and efficient way to solve this problem is to rely on linear programming. Indeed, the problem can be stated using only linear constraints and a linear objective function, i.e.

$$\begin{aligned} \text{Objective:} \quad & min \sum_{i=1}^{13} |F_i|/F_{max}^i \\ \text{Constraints:} \quad & (i)\ 0 \leq F_i \leq F_i^{max}\ (i = 1...13) \\ & (ii)\ \tau = R \cdot F \end{aligned}$$

Normalizing each force by its maximum force is expected to distribute the forces according to the muscle capacities, i.e. in a way similar to what humans would do. The linear programming toolbox from the GLPK library (GNU Linear Programming Kit [17]) was exploited to find the unique optimal solution, i.e. the reference muscle forces.

2) Finding the muscular stimulations: The next step is, for each muscle, to compute the neural stimulations corresponding to this reference force. The following paragraph details the steps required for this computation. The procedure is closely linked to the direct model equations from [1].

From the current skeletal configuration, the first step consists in computing all muscles lengths l_{mtu}. The second step consists (ii) in extracting the series element length based on the desired muscular force. By definition, $F_{se} = F$, so that,

$$l_{se} = \epsilon_{ref} l_{slack} \sqrt{F/F_{max}} + l_{slack} \tag{3}$$

with F being the desired force, F_{max} the muscle maximum force, l_{slack} the slack length (the series elastic element length under which the buffer element engages) and ϵ_{ref} the muscle reference strain. Then (iii), as depicted in Fig. 2, the contractile element length can be extracted using the total muscle length l_{mtu}:

$$l_{ce} = l_{mtu} - l_{se} \tag{4}$$

The following step (iv) involves a time differentiation, required to obtain the contractile velocity v_{ce}. Indeed, in the direct model, the contractile length l_{ce} is obtained by integrating the contractile velocity. This differentiation is implemented using a backward finite differences scheme of order three. Then, the internal forces and force/length and force/velocity relationships are computed (v) following the direct model. The last step (vi) requires to invert the force relationship, bounding the result in the simulation interval, which is $[0.01; 1]$, the lower bound being the basal activity, i.e. the minimum muscular activity. This provides the muscular stimulation s that corresponds to the desired force F:

$$s = \frac{F_{ce}}{F_{max} f_l f_v} \tag{5}$$

Last but not least, we define the reconstruction error e_r as the difference between the reference torques generated by the impedance controller and the torques generated by the musculoskeletal model when stimulated with the output of the inversion module. This reconstruction error is central in evaluating the performances of the inversion module.

III. VALIDATION TOOLS & PROTOCOLS

The proposed algorithms were validated by controlling a simulation model of the 95 cm tall COmpliant HuMANoid platform (COMAN, see Fig. 4). This robot, developed by the Italian Institute of Technology (IIT), has 23 actuated degrees of freedom (DOFs), most of them being equipped with series elastic actuators [18], [19] and [20]. Each joint is equipped with position, velocity and torque sensors. The robot also features an inertial measurement unit (IMU) and 6-DOF feet sensors measuring ground reaction forces and torques. Our controller only uses the sensory inputs available on the actual robot. The COMAN was modeled in a simulation environment called Robotran [21]. This simulator provides a direct dynamics engine for rigid multi-body systems. An accurate modeling of the robots series elastic joints was implemented and described in [18]. For all the following experiments, the robot joints are torque driven using the lower-level controller available on the real robot. As depicted in Fig. 1, only the lower limbs are driven by the musculoskeletal model while the other joints are systematically driven by the impedance controller.

In a first experiment, the muscular inversion module was validated by comparing the reference and reconstructed torques, i.e. the torques produced by the impedance controller (Fig. 1 (c)) and the torques generated by the musculoskeletal model being driven by the stimulations from the inversion module (Fig. 1 (b)). The simulation started with the robot standing straight with both feet aligned as depicted in Fig. 4, rightmost panel. Three perturbations were applied on the robot body, each lasting 0.2s and separated by 3s from each other (see Fig. 4 for the forces application points): a 25N force on the robot trunk, horizontally in the sagittal plane and in the forward direction; a 15N force on the robot waist, horizontally in the frontal plane, lateral direction; and a 20N force on the robot wrist, horizontally in the forward direction. The signals were recorded with the impedance controller governing the robot reactions to perturbations (i.e. no learning in this case).

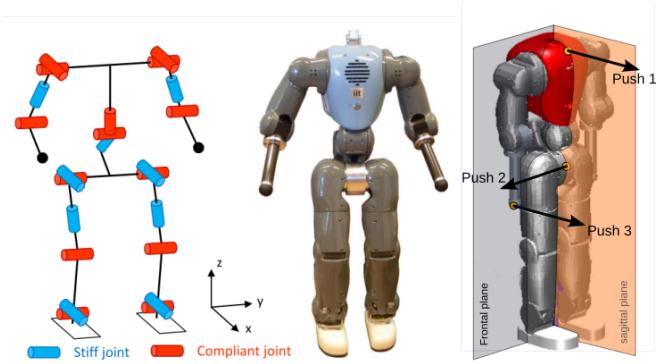

Fig. 4. COMAN: a 23 DOF compliant humanoid robot: real robot and simulation model (adapted from [3]).

In a second experiment, the learning performance was validated by recording the evolution of the cognitive control ratio (see section II-C). The threshold error on the predicted stimulations that triggers learning (cognitive control active) was fixed to 0.04. Table II summarizes the machine learning parameters used for this experiment. The simulation again started with the robot standing straight with both feet aligned as depicted in Fig. 4. The validation was performed through two different perturbation scenarios. In the first one, perturbations were restricted to the sagittal plane. The robot underwent a force perturbation on the torso with a random magnitude in the range $[-10, 30]$N, every 4s and lasting 0.2s each. A qualitative validation of the learning progress was also provided for this scenario. The reference and predicted stimulations were recorded at different stages of the learning process for the same 10N push perturbation on the torso and in the sagittal plane (0, 5 and 50 pushes). As more data becomes available, learning should progress up to making the robot able to predict the stimulations corresponding to the sensory information arising from the perturbed posture. In the second scenario, learning in the other planes (frontal and transverse) was validated. The robot underwent a 3D horizontal force perturbation on the trunk with a random amplitude ("Push 1" application point in Fig. 4). The sagittal and frontal components are uniformly selected in the ranges $[-5, 15]$N and $[-10, 10]$N respectively. The duration was fixed to 0.2s and the perturbation are triggered every 4s. For both scenarios, five runs were performed. Both CMAC and SVR were tested and compared in the first scenario, while the

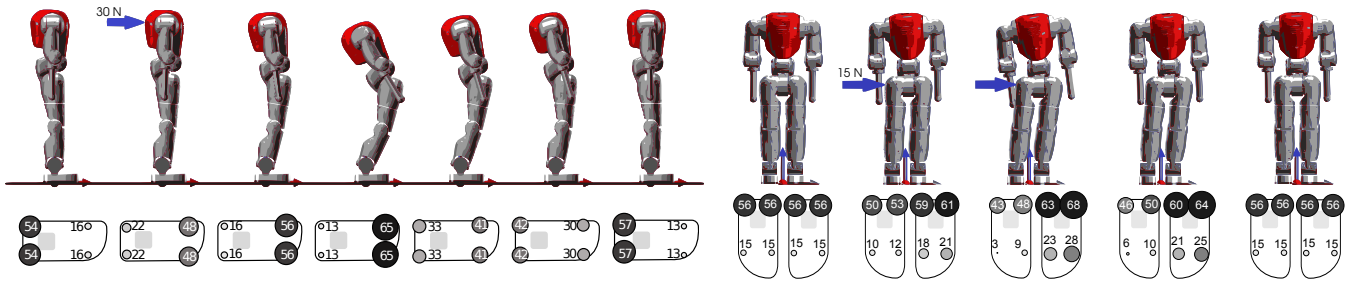

Fig. 5. Recovery motion for a horizontal perturbation on the torso (sagittal plane) and a horizontal perturbation on the waist (frontal plane). The lower panels show the evolution of the contact forces (in N) during the recovery motion.

TABLE II
MACHINE LEARNING PARAMETERS

CMAC			SVR			
LU size	Quant.	#AU	C	γ	ϵ	K
400000	300	10	200	1	0.02	RBF

3D scenario was only tested with SVR due to the relatively poorer performance of CMAC in 2D (see Section IV). SVR offline retraining occurred every 15s or earlier if more than 5000 stimulation errors exceeding the threshold (across all muscles) have been detected since the last retraining.

IV. RESULTS

The quality of the generated reference stimulations (muscular inversion) and the ability of the robot to learn these stimulations were tested in the simulation environment following the previously reported protocol. Fig. 5 shows the robot recovery motion for a sagittal and a frontal perturbation with the neural controller governing the robot motion. The force under the four virtual foot contact points show how the robot transferred the position of his CoP to stabilize the motion of the CoM.

A. Inverse muscular reconstruction

Fig. 6 shows the 13 muscular stimulations for the right leg that were generated for the first experiment (three pushes). A noticeable element is that the gluteus muscle (GLU, see Fig. 2) was never recruited by the optimization to counter the perturbations. For the same set of perturbations, Fig. 7 shows the reference and reconstructed torques for the six degrees of freedom from the right leg when using the inverse muscular model. The mean squared error (mse) for each joint is computed.

B. Cumulative learning

Fig. 8 gives a qualitative insight into the learning process for the sagittal muscles. As more experience was accumulated, the predicted stimulations (from the SVR regression engine) became more accurate and converged towards the reference stimulations (from the muscular inversion module). At the beginning of the learning phase, the predicted stimulations were systematically zero. At the end of the learning phase, the residual error was systematically below

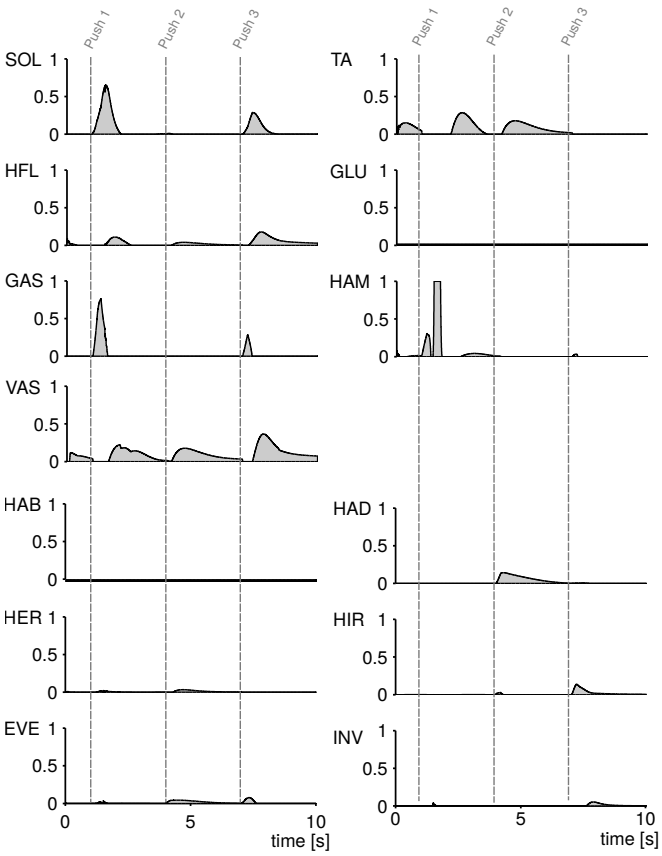

Fig. 6. Muscular stimulations for all 13 muscles generated by the inverse muscle model when the robot is receiving the benchmark perturbations.

the learning threshold. Fig. 9 shows the global results of the second experiment, first scenario. In particular, it shows that the control was gradually transferred from the impedance controller to the neural controller as more training data was available. After 160 pushes in the sagittal plane, while SVR managed to fully control the robot, CMAC can only predict accurate stimulations during around 75% of time. In the case of learning with 3D horizontal perturbations (second scenario), SVR also managed to reduce the cognitive control ratio to a low value, as depicted in Fig. 10. A video illustrating these results is provided as a supplementary material.

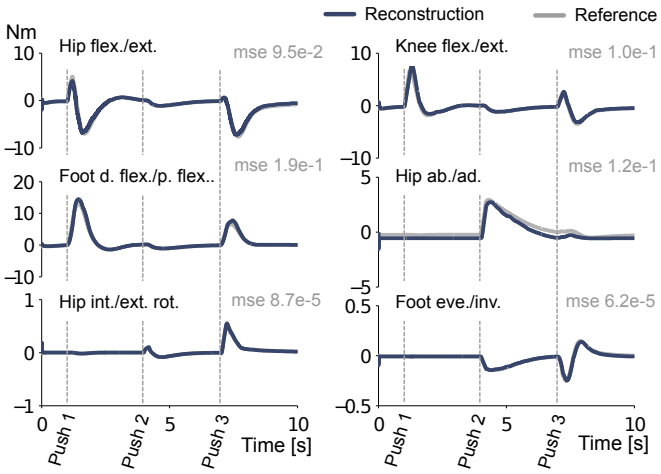

Fig. 7. The reconstructed torques (dark blue) approximate with high accuracy the reference torques (grey).

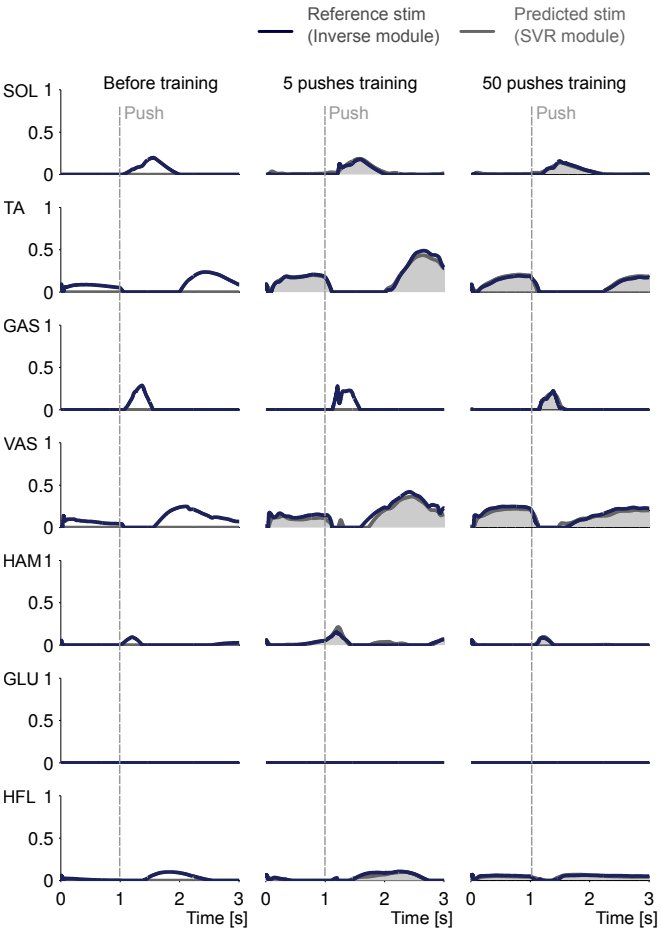

Fig. 8. Evolution of the predicted muscular stimulations at different learning stages for the same forward push, in the sagittal plane and on the torso (10N). The robot is trained with random magnitude pushes in the sagittal plane.

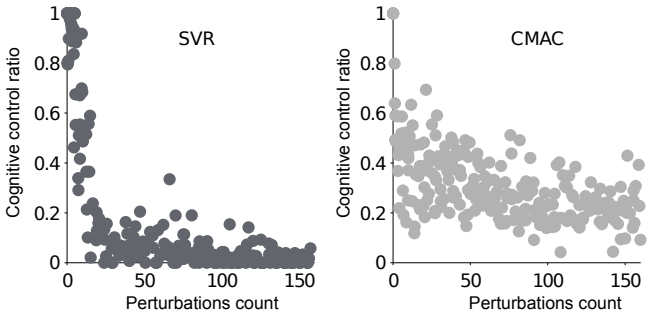

Fig. 9. The cognitive control ratio decreases as more training data becomes available. The figure overlays five runs while delivering perturbations in the sagittal plane only.

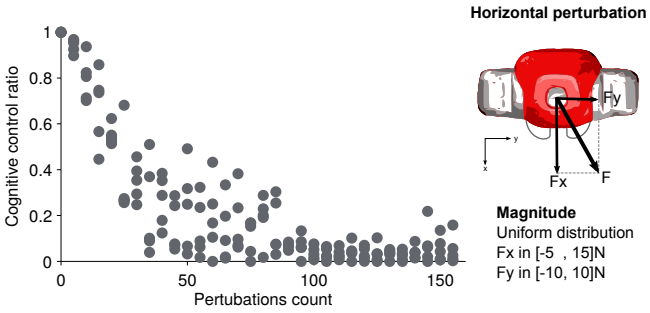

Fig. 10. Using SVR, the cognitive control ratio decreases as more training data becomes available. The figure overlays five runs while delivering perturbations in the whole transverse plane.

V. Discussion & Perspectives

The previous section provided quantitative results about the ability of the robot to predict muscular stimulations necessary to keep balance. The torques reconstruction of Fig. 7 shows an excellent performance for computing adequate muscular stimulations. The small reconstruction error has presumably different causes: (i) the limited accuracy of the numerical derivative necessary to compute the muscles contraction velocity, (ii) the basal muscular stimulation that generates parasitic forces (muscular stimulations are bounded between $[0.01, 1]$ with the lower bound being the basal activity) and (iii) the muscular saturation that limits the muscles output force (muscular stimulations saturate to an upper bound). The systematic silence of the gluteus muscle (GLU) can be explained by the fact that the biarticular hamstring muscle (HAM), also acting on the sagittal hip, can generate the required torque alone.

We showed that the neural controller is able to learn the stimulations with high accuracy, leading to an almost exclusive control by the neural controller (i.e. with no muscular model inversion) even for 3D random perturbations. The residual error is probably due to the dimensional reduction of the problem (SVR/CMAC engines only receive a subset of all available sensory inputs to limit the nonlinear growth of the model as a function of the dimension of the input space). With a longer learning period, the robot would presumably further reduce the cognitive control ratio to lower values, the bottleneck being the richness of sensory inputs. We also

highlighted the better performance of SVR as compared to CMAC for predicting stimulations when using the given settings. Moreover, the neural prediction is fast compared to the impedance computation (even faster with dedicated hardware) which is important for real-time control. This is highly desirable for future testing on real robots.

While muscles display interesting visco-elastic properties that have perturbation filtering capabilities, they also enable the coordination between joints due to the presence of multi-articular muscles with variable stiffness. In the line of this paper, these properties can be more thoroughly explored to improve the controller robustness. Furthermore, a reinforcement learning technique might outperform the actual performances obtained using supervised learning from the impedance controller. Also, the analysis of the learned stimulations might show some correlations with the sensory inputs, enabling the synthesis of simple neural control rules.

The development of the inverse muscular model is valuable to better understand the actual human behavior when subject to perturbations. For instance, the stimulations computed by this algorithm can be later compared to EMG signals measured on humans. Or following the same bio-inspired approach, data acquired during human balancing experiments could also be used for learning, taking advantage of biological optimisations. On top of that, the proposed algorithm manages push recovery by computing virtual muscle stimulations. Other algorithms using stimulations-driven neuromuscular models (like the walking controllers of [2] and [8]) already produce energy-efficient human-like gaits. Incrementing them with the algorithms developed in this paper would offer to use the same tools to produce virtual muscle stimulations both for walking and for robust postural control.

Finally, future work will focus on extending this controller to counter large perturbations for which a stepping motion is required.

VI. Conclusion

The developed neuromusculoskeletal model displayed the ability to generate muscular stimulations to counter external perturbations in order to keep balance control of a humanoid robot. Two machine learning techniques, namely CMAC and SVR, were applied for the generation of a regression model capturing the muscular stimulations required for balance. An impedance controller regulating the CoM position and a module inverting the Hill-muscle model were also developed to produce reference stimulations required during the learning process. The algorithm was tested in simulation with a torque-controlled child-sized humanoid robot (COMAN). The results suggest that the control can be gradually transferred from the impedance/inverse model (training modules) to the regression models (neural controller), as learning progresses. This approach was validated by applying random perturbations forces in the sagittal plane and in the whole transverse plane. The robot managed to learn the correct stimulations required to withstand small to medium perturbations, i.e. with no stepping being necessary.

References

[1] H. Geyer and H. Herr, "A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities." *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 18, no. 3, pp. 263–73, June 2010.

[2] N. Van der Noot, A. J. Ijspeert, and R. Ronsse, "Biped gait controller for large speed variations, combining reflexes and a central pattern generator in a neuromuscular model," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6267–6274.

[3] N. Van der Noot, L. Colasanto, A. Barrea, J. van den Kieboom, R. Ronsse, and A. J. Ijspeert, "Experimental validation of a bio-inspired controller for dynamic walking with a humanoid robot," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 393–400.

[4] B. Stephens, "Humanoid push recovery," *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 589–595, Nov. 2007.

[5] J. S. Albus, "A theory of cerebellar function," *Mathematical Biosciences*, vol. 10, no. 12, pp. 25 – 61, 1971.

[6] Z. Li, N. G. Tsagarakis, and D. G. Caldwell, "A passivity based admittance control for stabilizing the compliant humanoid COMAN," *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 43–49, Nov. 2012.

[7] S.-h. Hyon, J. G. Hale, and G. Cheng, "Full-Body Compliant Human-Humanoid Interaction: Balancing in the Presence of Unknown External Forces," vol. 23, no. 5, pp. 884–898, 2007.

[8] S. Song and H. Geyer, "Generalization of a muscle-reflex control model to 3d walking." *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2013, pp. 7463–6, Jan. 2013.

[9] A. Hill, "The heat of shortening and the dynamic constants of muscle," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 126, no. 843, pp. 136–195, 1938.

[10] E. M. Arnold, S. R. Ward, R. L. Lieber, and S. L. Delp, "A model of the lower limb for analysis of human movement." *Annals of biomedical engineering*, vol. 38, no. 2, pp. 269–79, Feb. 2010.

[11] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "OpenSim: open-source software to create and analyze dynamic simulations of movement." *IEEE transactions on bio-medical engineering*, vol. 54, no. 11, pp. 1940–50, Nov. 2007.

[12] A. Bejan and J. H. Marden, "Unifying constructal theory for scale effects in running, swimming and flying." *The Journal of experimental biology*, vol. 209, no. Pt 2, pp. 238–48, Jan. 2006.

[13] A. Schepelmann, M. D. Taylor, and H. Geyer, "Development of a Testbed for Robotic Neuromuscular Controllers," in *Robotics: Science and Systems*. MIT Press, 2012.

[14] A. Smola and B. Schlkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[15] R. L. Smith, "Intelligent Motion Control with an Artificial Cerebellum," no. July, 1998.

[16] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.

[17] "Gnu linear programming kit, version 4.55." http://www.gnu.org/software/glpk/glpk.html

[18] H. Dallali, M. Mosadeghzad, G. a. Medrano-Cerda, N. Docquier, P. Kormushev, N. Tsagarakis, Z. Li, and D. Caldwell, "Development of a dynamic simulator for a compliant humanoid robot based on a symbolic multibody approach," in *2013 IEEE International Conference on Mechatronics, ICM 2013*. IEEE, Feb. 2013, pp. 598–603.

[19] N. Tsagarakis, M. Laffranchi, B. Vanderborght, and D. Caldwell, "A compact soft actuator unit for small scale human friendly robots," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 4356–4362.

[20] N. G. Tsagarakis, S. Morfey, G. Medrano Cerda, Z. Li, and D. G. Caldwell, "COMpliant huMANoid COMAN: Optimal joint stiffness tuning for modal frequency control," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 673–678.

[21] Robotran, "Modeling Multibody Systems with ROBOTRAN," Université catholique de Louvain, Louvain-la-Neuve, Tech. Rep., 2009.