A Scalable Neuro-inspired Robot Controller Integrating a Machine Learning Algorithm and a Spiking Cerebellar-like Network

Baira Ojeda, Ismael; Tolu, Silvia; Lund, Henrik Hautop

Published in:
Proceedings of Living Machines 2017

Link to article, DOI:
10.1007/978-3-319-63537-8 31

Publication date:
2017

Document Version
Peer reviewed version

A Scalable Neuro-inspired Robot Controller Integrating a Machine Learning Algorithm and a Spiking Cerebellar-like Network

Ismael Baira Ojeda[1,*]✉, Silvia Tolu[1,*] and Henrik H. Lund[1]

Technical University of Denmark, Elektrovej Building 326, DK-2800 Kgs. Lyngby, Denmark,
{iboj, stolu, hhl}@elektro.dtu.dk[⋆]

Abstract. Combining Fable robot, a modular robot, with a neuroinspired controller, we present the proof of principle of a system that can scale to several neurally controlled compliant modules. The motor control and learning of a robot module are carried out by a Unit Learning Machine (ULM) that embeds the Locally Weighted Projection Regression algorithm (LWPR) and a spiking cerebellar-like microcircuit. The LWPR guarantees both an optimized representation of the input space and the learning of the dynamic internal model (IM) of the robot. However, the cerebellar-like sub-circuit integrates LWPR input-driven contributions to deliver accurate corrective commands to the global IM. This article extends the earlier work by including the Deep Cerebellar Nuclei (DCN) and by reproducing the Purkinje and the DCN layers using a spiking neural network (SNN) implemented on the neuromorphic SpiNNaker platform. The performance and robustness outcomes from the real robot tests are promising for neural control scalability.

Keywords: Neuro-robotics, bio-inspiration, motor control, cerebellum, machine learning, compliant control, internal model

1 Introduction

The brain carries out tasks in a formidable manner, smoothly and with a low power consumption. In contrast, robots lack the adaptability and precision of human beings. Thus, our main interest is to study how the central nervous system (CNS) controls the human body, coordinates smooth movements and the mechanisms of motor control and motor learning towards the development of bio-inspired autonomous robotic systems [1]. We present a bio-inspired closed-loop control architecture that integrates a machine learning (ML) technique and a cerebellar-like microcircuit to provide real-time neural control. The result is a compliant system for motor learning that can scale to several neurally controlled robot modules. This is achieved by combining a modular robot with the neuromorphic computing platform, SpiNNaker [2]. The ML engine is the Locally Weighted Projection Regression algorithm (LWPR) [3].

[⋆] *These authors contributed equally to this work.

1.1 The Cerebellar Microcircuit

The presented bio-inspired approach aims at emulating the cerebellar function in learning and modulating accurate, complex and coordinated movements. The cerebellum is involved in the neural control of bodily functions [4], such as postural positioning, balance or coordination of movements over time. Research studies [5], [6], [7] described the cerebellum as a set of adaptive modules, also called cerebellar microcomplexes, embedded in the motor control system to improve coordinated movements over time. Ito [5] stated that each microcomplex is a Unit Learning Machine (ULM): a set of structured neural circuits that modify the relationship between the input and output in response to error signals. The ULM encodes the internal model (IM) in order to precisely perform the control of the body part without referring to feedback. At the core of our control architecture, a cerebellum-like model is implemented akin to the Marr-Albus cerebellum model [8], [9]. The cerebellar cortex comprises three layered sheets wrapped around the deep cerebellar nuclei (DCN) and it connects to the brain stem via the superior, middle and inferior peduncles. The microcomplexes that correspond to the minimal functional unit, show a similar internal microcircuitry illustrated in Fig. 1. The main inputs are: the mossy fibers (MFs) and the climbing fibers (CFs). MFs transmit sensory information originated from multiple extra-cerebellar sources and carry out excitatory synapses with the dendrites of the granule cells (GCs) and the deep cerebellar nuclei cells (DCNs). CFs arise exclusively from the inferior olive (IO) within the brain stem and they are particularly responsive to unexpected somatosensory events. Thus, they are usually conceived as error signal transmitters. The output of the cerebellar cortex is performed by the Purkinje cell (PCs), which combines information received from the CFs and the parallel fibers (PFs). The PC produces a "complex spike" response [10] each time it receives input from a CF. Finally, the DCNs combine the inhibitory signal coming from the PCs (cerebellar cortex output) with the excitatory information coming from both CFs and MFs. The PC and DCN layers are modeled in SpiNNaker by a Spiking Neural Network (SNN) that, instead of receiving inputs by means of MFs, receives pre-processed signals from the LWPR receptive fields (RFs) (see Fig. 1).

1.2 Related Work

Cerebellar mechanisms of motor learning and control are not yet completely unveiled [11]. By mimicking them, researchers are eager to uncover those unknowns and at the same time to develop scalable and useful neural control techniques. Indeed, some techniques have already benefited robotics such as successful simulations where robots are controlled using biologically plausible cerebellar models [12], [13], [14], virtual experiments with a simulated robot using customized brain models, e.g. the Neurorobotics Platform (NRP) [15], bio-inspired abstraction strategies of robotic dynamics and kinematics [16], and adaptive control schemes for non-linear systems [17], [18], [19]. Besides, realistic cerebellar SNN

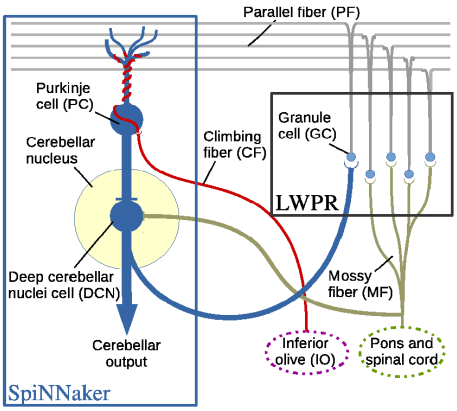

Fig. 1: The simplification of the cerebellum's internal circuit. On the left, the sub-circuit reproduced using SpiNNaker. On the right, the granular layer represented by the LWPR algorithm.

implementations on a computer [20], e.g. [21] and on the neuromorphic platform, Spinnaker [2], e.g. [22] to control a neurorobot are providing promising results.

Different neuromorphic hardware platforms have been developed, but most of them lack in terms of scalability and usability (FPGAs,GPs) [22]. To this end, the computer is still the prevailing architecture for neural simulations of non-large scale neural networks. In this article we present the unique proof-of-concept system combining SNN and ML technique for controlling a modular robot by CPU-Spinnaker co-processing.

1.3 Objectives and Paper Organization

Our research area has three main goals: to embed a strategy for a fast learning of the inverse IMs of the robot module providing an optimized input representation to the SNN. Indeed, the LWPR algorithm was chosen because it encodes the inputs like the cerebellar GCs expansively; to develop a neural control for mimicking the cerebellar modularity, i.e. the neural core can be replicated and dedicated to a specific robot part or module. Both SpiNNaker and the LWPR algorithm allow this internal scalability. The first in terms of neural cores and the latter in terms of incremental locally linear models; and to develop a user-friendly system in which the robot topology can be easily modified and controlled. The present paper addresses the first goal and sets out the basis by which the other two goals are going to be achieved in the future.

The outline for the next sections is as follows: Section 2.1 and 2.2 describe the robot employed and Spinnaker; Section 2.3 addresses the blocks of the adaptive control architecture; Section 3 explains the interface with SpiNNaker; Section 4 and 5 show the results giving a discussion and conclusions.

2 Material and Methods

2.1 Fable Robot

Fable [23] is a modular robot based on self-contained 2-DoF modules. It consists of multiple simple robotic modules that can attach and detach. Connectors between units allow the creation of arbitrary and changing structures depending on the task to be solved. Modules are easily addressed by an ID and can be programmed at different levels of abstraction. Fig. 2 illustrates the Fables network architecture. The joint commands are transmitted from the computer using a radio dongle. The system is scalable to more than one Fable module by simply creating an object instance and indicating the IDs of the modules to take into account. The Spinnaker board described in the next section paves the way for large-scale modular neuro-robotic systems.

2.2 Neuromorphic Platform

SpiNNaker [2] is a novel chip based on the ARM processor that was designed to support large scale spiking neural networks simulations. Inspired by the structure of the brain, the processing cores were arranged in independently functional and identical Chip-Multiprocessors (CMP) to achieve robust distributed computing. Each processing core is self-sufficient in the storage required to hold the code and the neurons states, while the chip holds sufficient memory to contain synaptic information for the neurons in the system connected to the local neurons.

2.3 Adaptive Control Architecture

Our control approach is based on the Adaptive Feedback Error Learning (AFEL) architecture described in [16]. In this work, we included the DCN (see Fig. 3).

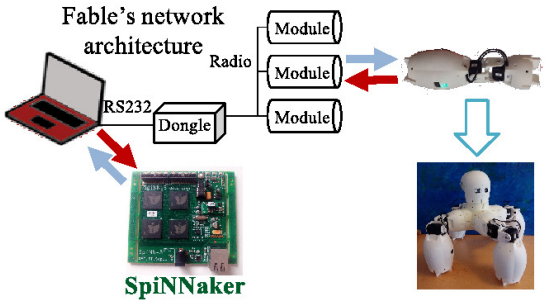

Fig. 2: Fable robot is equipped with Dynamixel AX-12A motors. The ULM is embedded in the control architecture to control a single Fable module. The LWPR engine is run in the computer and sends the inputs to the SNN implemented in SpiNNaker. The cerebellar output from SpiNNaker is the joint motor command, which is transmitted from the computer to Fable motor using a radio dongle. Fable modules can be snapped together towards different topologies.

The architecture comprises: a trajectory planner which computes the desired joint angles and velocities $(Q_{dj}, \dot{Q_{dj}})$ by inverse kinematics; a Learning Feedback (LF) controller which generates the τ_{LF_j} feedback joint torques; and a ULM, which provides the τ_{ff_j} feed-forward joint torques. The τ_{ff_j} torque contribution of the ULM is a combination of the τ_{LWPR_j} prediction from the LWPR algorithm and the τ_{c_j} prediction from the PC-DCN layers for each joint j (see Fig. 3). In addition, the τ_{tot_j} global torque is the summation of the τ_{ff_j} and the τ_{LF_j} command joint torques.

If the adaptive model is accurate, the resulting τ_{ff} cancels the robot nonlinearities. However, if the inverse dynamic model is not exact, the LF reflects the feedback error between the desired signal $(Q_d, \dot{Q}_d)$ and the output of the real robot $(Q, \dot{Q})$. The LWPR incrementally learns the inverse IM of the robot module from the τ_{tot} global torques or efferent copy. The LWPR produces τ_{LWPR} torques to minimise the τ_{LF} (error related estimate), while the PC-DCN layer provides τ_C corrective torques which refine the τ_{tot} global command torques. Three plastic sites (PF-PC, PC-DCN, MF-DCN) are represented by the PC-DCN layers, whose learning benefits from a compact sensorimotor representation of the input space [17], [24] by means of p_k weights provided by the LWPR algorithm. See below for implementation details.

ULM Inside the Unit Learning Machine, the LWPR algorithm feeds the sensorimotor inputs to N linear local models. The x_i inputs consist of cerebellar

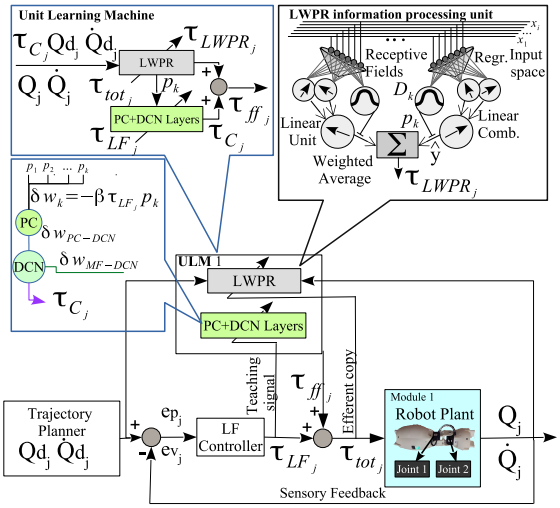

Fig. 3: The AFEL control architecture embeds a cerebellum-like model which acts as a feed-forward controller in the form of ULM. Even though only one ULM was used to control one Fable module, the system can be scaled up to control more modules by embedding several ULMs at the same time.

torque commands, desired and current positions and velocities of every joint j. The LWPR incrementally divides the input space into a set of RFs, so that a weighting kernel computes a weight p_k for each x_i data point according to the distance from the c_k center of the kernel in each k local unit. The weight is a measure of how often an item of x_i data falls into the region of validity of each linear model. The weights are calculated using a Gaussian kernel (1):

$$p_k = \exp\Big(-\frac{1}{2}(x_i - c_k)^T D_k (x_i - c_k)\Big), \tag{1}$$

where D_k is a positive definite matrix which is called distance matrix (the size of the RF). This measure is updated on-line iteratively by using an incremental gradient descent based on stochastic leave-one-out cross validation criterion. In every iteration, the RF weight is updated in order to assign the new inputs to the closest RF. The LWPR output is the weighted mean of the linear models:

$$\hat{y} = \frac{\sum_{k=1}^{N} p_k \hat{y}_k}{\sum_{k=1}^{N} p_k} \tag{2}$$

In this work, $\hat{y}_k$ is the τ_{LWPR} torque shown in Fig. 3. The LWPR learns the τ_{tot} efferent copy. As it occurs in the cerebellum, the PC-DCN inputs are transmitted through a bank of filters, located in the GC layer. In our approach, this bank of filters is represented by the the $p_k(t)$ signals computed by the gaussian kernel in (1). Those p_k weights are driven to the synapses with the PC layer. The data flow along the SNN is illustrated in Fig. 4. First, the p_k input weights coming from the LWPR algorithm are transformed into spikes. The spiking rate of each PF is defined as one spike every T_i ms. The spikes are transmitted along the spiking network consisting of one pair of PCs and DCNs for each joint. Finally, the DCN output is calculated as inversely proportional to the DCN spike frequency rate. We selected the Leaky Integrate and Fire model with fixed threshold and decaying-exponential post-synaptic current as the neuron model since it showed smooth trajectories and fast computations. To apply memory consolidation in the SNN, we chose the Spike-Timing-Dependent plasticity (STDP) [25] whose synaptic learning is induced by tight temporal correlation between a pre- and a post-synaptic spike event.

LF Controller The Learning Feedback controller overcomes the lack of a precise robot arm dynamic model, ensures the stability of the system and enables the control architecture to achieve a better performance [16]. Further details about the LF controller are provided in [16]. Its gains were tuned to $K_p = 7.5$, $K_v = 6.4$ and $K_i = 0.22$ for the Fable robot.

3 Interface

PyNN language [26] was selected to implement the SNN since it can run on a number of simulators with minor modifications of the code. A socket interface

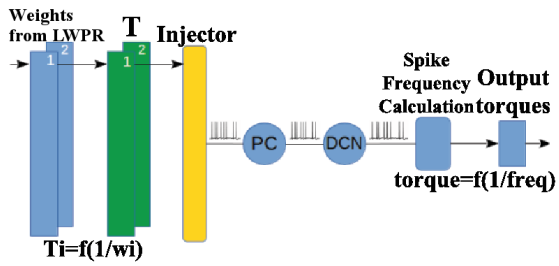

Fig. 4: The encoding and decoding principle behind the interface with SpiNNaker. The p_k weights coming from the LWPR are encoded into spiking rates so as to excite the population. Thereafter, the DCN spiking rate is transformed into torques.

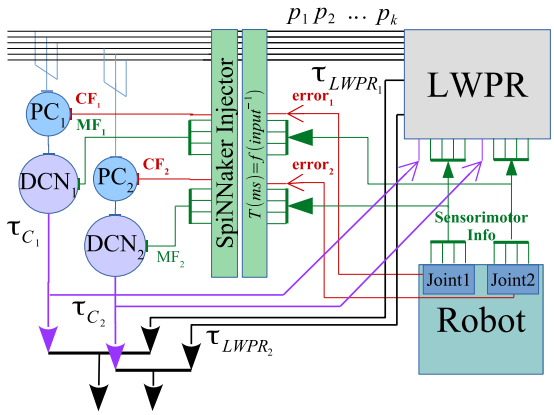

Fig. 5: Schematic of the connections needed for the integration between the LWPR, SpiNNaker and the robot.

combined with thread functions was implemented to allow the communication between SpiNNaker and Fable robot. On the one hand, the socket interface enables the system to push and read data on-line between scripts that are running different python versions. The interface updates p_k weights every 5ms and the control loop frequency is 150Hz. On the other hand, the firing rate is calculated to deliver the cerebellar output to the robot.

4 Evaluation

To evaluate the control system performance, we examined how the tracking errors became compensated during the task of following the desired trajectory defined in (3), where Q_j is the angle of the j-th joint, A is the amplitude of the circular trajectory, and C_j was set to 0 and $\pi/2$ for joints 1 and 2, respectively.

$$Q_j = A\left(\frac{1}{2\pi f}\right)^2 sin(2\pi f t + C_j), \tag{3}$$

4.1 Control performance

We measured the learning performance of the system in terms of the normalized mean squared error (nMSE). The error was considered as the difference between the desired and real position of each joint during the experiment. To this end, 20k iterations of the control loop, equivalent to 200 circles iterations, were run commanding the real robot to trace out a circular trajectory with its end-effector.

Fig. 6 shows how the nMSE is decreasing over consecutive iterations. It is remarkable the fast adaptation. Furthermore we evaluated the torque contribution along the learning process by the performance ratios of torque components to the τ_{tot} global torque applied to the robot module plant. Fig. 7 shows how the average values of the ratios evolve along the learning process. The LWPR algorithm progressively learns the τ_{tot} global torque which means that it acts as inverse IM for the robot module dynamics. LF torques decrease over time according to the decrease of the nMSE, and the contribution from the PC-DCN layers reveals the slow memory consolidation function which is achieved within the LWPR and more slowly achieved in the PC-DCN layers as shown in logarithmic scale.

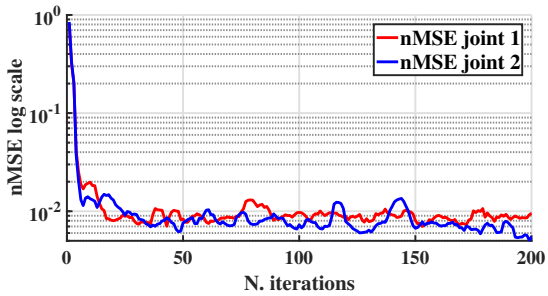

Fig. 6: Performance test. The performance of our neuro-controller during real-time simulation on computer and SpiNNaker platform in terms of the nMSE.

In order to check the advantages of the presented approach, we compared it with other three cases enumerated in Fig. 8 with labels (2), (3) and (4). In case (2) only the LF was active. In case (3) the PC-DCN layer was deactivated. In case (4) the LWPR was learning but not delivering its torque to the control loop. Notice the large tracking error of case (2) and how the PC-DCN contribution (case 4) leads to a faster decrease of the nMSE compared with case (3). It is worth noting that the improvement of our approach is due to the addition of the DCN layer, and not due to the integration of SpiNNaker. Nevertheless, we integrated SpiNnaker aiming at scaling up the spiking cerebellar model in the

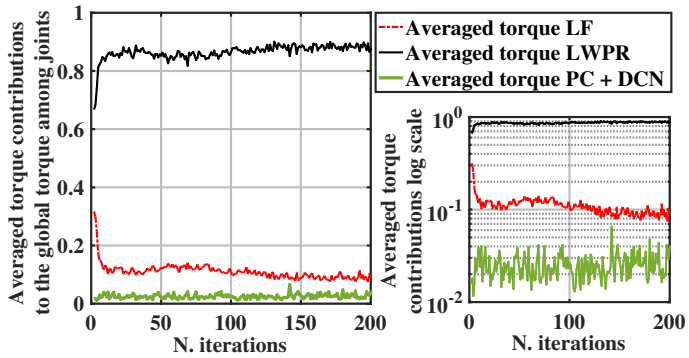

Fig. 7: Performance test. Ratios of torque contributions of the main blocks to the global torque averaged among joints.

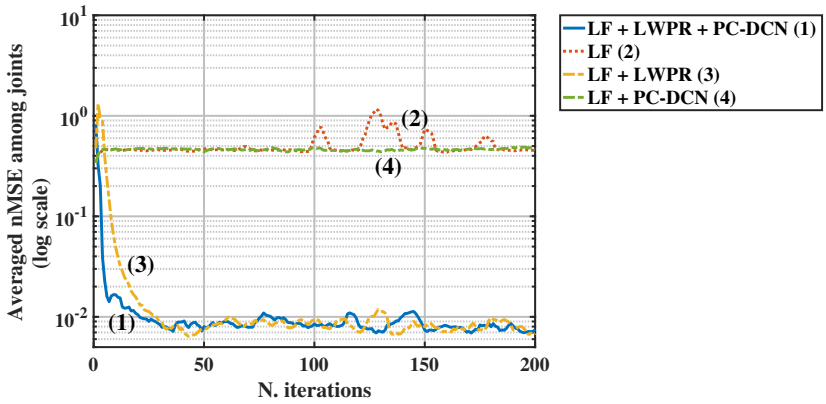

Fig. 8: Comparison of the averaged nMSE among joints for four different cases.

future. SpiNNaker will be highly beneficial when implementing large amounts of spiking neurons and synapses in real time. However, CPU-based system may show issues in the performance and power consumption.

4.2 Robustness tests

The robustness test consisted in switching the amplitude of the desired trajectory every 15 repetitions of the circle trajectory. Fig. 9 shows that the approach is robust and self-adaptive to fast and repetitive changes. Furthermore, results in Fig. 10 indicate that the LWPR learning is incremental and so, the LF decreases its feedback torque contribution eventually. In the same way, the PC-DCN contribution depends on the LWPR algorithm due to its capability of incorporating the IM and to p_k signals, which correspond to the cerebellar granular weighting kernels.

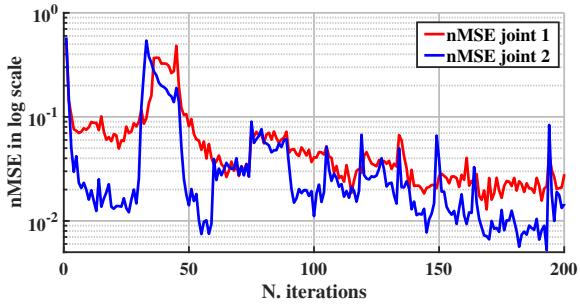

Fig. 9: Robustness test. nMSE result when forcing the system to relearn different amplitudes of the trajectory by switching the amplitude of the target trajectory every 15 circle iterations. The amplitude values were chosen randomly between $A \in \{500, 700, 900\} deg/s^2$. Higher nMSE correspond to higher amplitudes.

5 Conclusions

We proposed a new Adaptive Feedback Error Learning scheme for the motor control and learning of a 2-DoF Fable module. The neural control is still based on the combination of a ML engine and a cerebellar-like model to both optimize the input space representation and to abstract the inverse IM. However, in this work, the cerebellar-like component is integrated with a spiking DCN layer running on SpiNNaker. This work was validated on a physical robot. We overcame the interface issues between SpiNNaker and Fable allowing an on-line control. The presented robustness and performance results were good showing an increase of the learning speed of the inverse IM while providing an optimized input representation. We will also integrate distinct sensors and actuators to check if it benefits the control and learning of the system.

SpiNNaker was designed to simulate large, biologically realistic, spiking neural networks in real time. We took advantage of only a small portion of the capability of SpiNNaker. Nevertheless, this paper shows the proof of principle for future achievements in the bio-inspired control and learning of robots using SpiNNaker. Further research will be needed to control more than one module and to benefit from this modular control strategy. Assembling distinct modular configurations will allow researchers and developers to carry out a variety of tasks with minor modifications. To this aim, all robot modules could be linked to identical ULMs. In future, we will test the system controlling more complex modular setups and study how to learn and store multiple IMs of the robot to face new challenges by combining previous experiences.

Acknowledgments This work has received funding from the EU-H2020 Programme under the grant agreement n.720270 (Human Brain Project SGA1) and from the Marie Curie project n. 705100 (Biomodular). We are thankful to David Johan Christensen and Moisés Pacheco, CEO/CTO & Co-founders of Shape

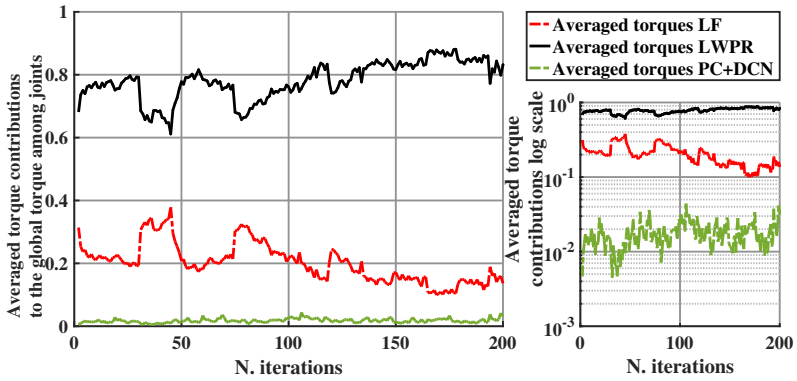

Fig. 10: Robustness test. Averaged torque contributions of the LWPR, LF and PC-DCN layers to the global torque when switching the amplitude of the target trajectory randomly ($A \in \{500, 700, 900\} deg/s^2$).

Robotics, for the Fable robot. A special thank is expressed to the University of Manchester and the University of Munich for SpiNNaker.

References

1. T. Flash and T. J. Sejnowski, "Computational approaches to motor control," *Current Opinion in Neurobiology*, vol. 6, no. 11, pp. 655–662, 2001.
2. S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.
3. S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.
4. M. Ito, "Cerebellar circuitry as a neuronal machine," *Progress in neurobiology*, vol. 78, no. 3, pp. 272–303, 2006.
5. ——, "Control of mental activities by internal models in the cerebellum," *Nature Reviews Neuroscience*, vol. 9, no. 4, pp. 304–313, 2008.
6. P. Dean, J. Porrill, C.-F. Ekerot, and H. Jörntell, "The cerebellar microcircuit as an adaptive filter: experimental and computational evidence," *Nature Reviews Neuroscience*, vol. 11, no. 1, pp. 30–43, 2010.
7. S. O. Verduzco-Flores and R. C. O'Reilly, "How the credit assignment problems in motor control could be solved after the cerebellum predicts increases in error," *Frontiers in computational neuroscience*, vol. 9, 2015.
8. J. S. Albus, "A theory of cerebellar function," *Mathematical Biosciences*, vol. 10, no. 1, pp. 25–61, 1971.
9. D. Marr and W. T. Thach, "A theory of cerebellar cortex," in *From the Retina to the Neocortex*. Springer, 1991, pp. 11–50.
10. C. C. Bell and T. Kawasaki, "Relations among climbing fiber responses of nearby purkinje cells." *Journal of Neurophysiology*, vol. 35, no. 2, pp. 155–169, 1972. [Online]. Available: http://jn.physiology.org/content/35/2/155
11. K. V. Byadarhaly, M. C. Perdoor, and A. A. Minai, "A modular neural model of motor synergies," *Neural Networks*, vol. 32, pp. 96–108, 2012.

12. N. R. Luque, J. A. Garrido, R. R. Carrillo, O. J. Coenen, and E. Ros, "Cerebellar input configuration toward object model abstraction in manipulation tasks," *Neural Networks, IEEE Transactions on*, vol. 22, no. 8, pp. 1321–1328, 2011.
13. N. R. Luque, J. A. Garrido, R. R. Carrillo, S. Tolu, and E. Ros, "Adaptive cerebellar spiking model embedded in the control loop: context switching and robustness against noise," *International Journal of Neural Systems*, vol. 21, no. 05, pp. 385–401, 2011.
14. J. A. Garrido, N. R. Luque, E. DAngelo, and E. Ros, "Distributed cerebellar plasticity implements adaptable gain control in a manipulation task: a closed-loop robotic simulation," 2013.
15. L. Vannucci, A. Ambrosano, N. Cauli, U. Albanese, E. Falotico, S. Ulbrich, L. Pfotzer, G. Hinkel, O. Denninger, D. Peppicelli, L. Guyot, A. V. Arnim, S. Deser, P. Maier, R. Dillman, G. Klinker, P. Levi, A. Knoll, M. O. Gewaltig, and C. Laschi, "A visual tracking model implemented on the icub robot as a use case for a novel neurorobotic toolkit integrating brain and physics simulation," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 1179–1184.
16. S. Tolu, M. Vanegas, N. R. Luque, J. A. Garrido, and E. Ros, "Bio-inspired adaptive feedback error learning architecture for motor control," *Biol Cybern*, vol. 106, pp. 507–522, 2012.
17. J. Porrill and P. Dean, "Recurrent cerebellar loops simplify adaptive control of redundant and nonlinear motor systems," *Neural computation*, vol. 19, no. 1, pp. 170–193, 2007.
18. S. Tolu, M. Vanegas, J. A. Garrido, N. R. Luque, and E. Ros, "Adaptive and predictive control of a simulated robot arm," *International journal of neural systems*, vol. 23, no. 03, p. 1350010, 2013.
19. F. Su, J. Wang, B. Deng, X.-L. Wei, Y.-Y. Chen, C. Liu, and H.-Y. Li, "Adaptive control of parkinson039;s state based on a nonlinear computational model with unknown parameters," *International Journal of Neural Systems*, vol. 25, no. 01, p. 1450030, 2015.
20. T. Yamazaki and J. Igarashi, "Realtime cerebellum: A large-scale spiking network model of the cerebellum that runs in realtime using a graphics processing unit," *Neural Networks*, vol. 47, pp. 103–111, 2013.
21. C. Casellato, A. Antonietti, J. A. Garrido, R. R. Carrillo, N. R. Luque, E. Ros, A. Pedrocchi, and E. D'Angelo, "Adaptive robotic control driven by a versatile spiking cerebellar network," *PLOS ONE*, vol. 9, no. 11, pp. 1–17, 11 2014. [Online]. Available:
22. C. Richter, S. Jentzsch, R. Hostettler, J. A. Garrido, E. Ros, A. Knoll, F. Rohrbein, P. van der Smagt, and J. Conradt, "Musculoskeletal robots: Scalability in neural control," *IEEE Robotics Automation Magazine*, vol. 23, no. 4, pp. 128–137, Dec 2016.
23. M. Pacheco, R. Fogh, H. H. Lund, and D. J. Christensen, "Fable: A modular robot for students, makers and researchers," in *Proceedings of the IROS workshop on Modular and Swarm Systems: from Nature to Robotics*, 2014.
24. N. Schweighofer, K. Doya, and F. Lay, "Unsupervised learning of granule cell sparse codes enhances cerebellar adaptive control," *Neuroscience*, vol. 103, no. 1, pp. 35–50, 2001.
25. J. Sjöström and W. Gerstner, "Spike-timing dependent plasticity," *Spike-timing dependent plasticity*, p. 35, 2010.
26. A. Davison, D. Brüderle, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "Pynn: a common interface for neuronal network simulators," 2009.