Scaling-up Action Learning Neuro-controllers with GPUs

Martin Peniak and Angelo Cangelosi

Abstract— **Neural networks have been used in many different robot motor-control experiments, however, so far the complexity of these neuro-controllers have remained at the similar level. The focus of this paper is to demonstrate that it is possible to scale-up these neuro-robotic controllers with GPUs leading to richer, more realistic and more complex motor control.**

I. Introduction

HUMANS are able to acquire many skilled behaviors during their life-times. The learning of complex behaviours is achieved through a constant repetition of the same movements over and over, with certain components segmented into reusable elements known as *motor primitives*. These motor primitives are then flexibly reused and dynamically integrated into novel sequences of actions. For example, the action of lifting an object can be broken down into a combination of multiple motor primitives. Some motor primitives would be responsible for reaching the object, some for grasping it and some for lifting it. These primitives are represented in a general manner and should therefore be applicable to objects with different properties. This capacity is known as *generalisation*, which also refers to the ability to acquire motor tasks by different ways. This means that the learning of new motor tasks can be done by using any body effector, or simply by imagining the actual task itself (see for example [1]). In addition, one might want to reach for the object and throw it away, instead of lifting it up. Therefore these motor primitives need to be flexible in terms of their order within a particular action sequence. The amount of combinations of motor primitives grows exponentially with their number and the ability to exploit this repertoire of possible combinations of multiple motor primitives is known as *compositionality*. The hierarchically organised human motor control system is known to have the motor primitives implemented as low as at the spinal cord level whereas high-level planning and execution of motor actions takes place in the primary motor cortex (area M1). The human brain implements this hierarchy by exploitation of muscle synergies and parallel controllers. These have various degrees of complexity and sophistication that are able to address both the global aspects of the motor tasks as well as fine-tune control necessary for the tool use [2].

The flexibility of the motor control system allows humans to execute behavioural actions, dynamically set the end point and degrees of freedom used for next task while being able to quickly adapt to various disturbances. Fogassi et al. argue that the flexibility of choosing different effectors is crucial to adaptability and related to the existence of peripersonal space [3]. Pioneering experiments on adaptation to rotating artificial gravity environments led to the general belief that humans would not be able to adapt to rotating environments with angular velocities over around 3 to 4 rpm (see [4]). An important study conducted by Lackner and DiZio showed that this sensorimotor adaptation is possible even with angular velocities reaching 10 rpm [5]. The experimental results showed that this can be achieved by making the same movement repeatedly, which allows the neural system to estimate and compensate for the Coriolis forces generated by a moving reference plane. These studies are clearly demonstrating the robustness and the flexibility of the human motor control system, which is capable of exploiting the use of motor primitives in order to reach higher level goals.

The existence of motor primitives and their recombination into sequences of actions is supported by the biological observations of both humans and animals. Sakai et al. conducted experiments in visiomotor sequential learning and demonstrated that his subjects spontaneously segmented motor sequences into elementary movements [6]. Thoroughman and Shadmehr showed that the complex dynamics of reaching motion is achieved by flexibly combining motor primitives [7]. d'Avella et al. analysed the data recorded from electromyographic activity from 19 shoulder and arm muscles and concluded that: *"the complex spatiotemporal characteristics of the muscles patterns for reaching were captured by the combinations of a small number of components, suggesting that the mechanisms involved in the generation of the muscle patterns exploit this low dimensionality to simplify control"* ([8], p. 7791). Experiments conducted on animals are also consistent with these findings. For example, it has been shown that the electrical stimulation of primary motor and premotor cortex in monkeys triggers coordinated movements such as reaching and grasping [9]. Giszter et al. found that a frog's leg contains a finite number of modules organised as linearly combinable muscle synergies [10].

Several action learning models have been proposed that implement functional hierarchies via explicit hierarchical structure, as with the MOSAIC model [11] or the mixture of multiple Recurrent Neural Networks (RNN) expert systems [12]. In these models the motor primitives are represented through local low-level modules, whereas higher-level modules are in charge of recombining these primitives using extra mechanisms such as gate selection systems. These systems carry great potential benefits. For example, the learning of one module does not interfere with the learning of other modules. Moreover, with the adding of extra low-level modules, the number of acquirable motor primitives can

Martin Peniak and Angelo Cangelosi are with the Centre for Robotics and Neural Systems, Plymouth University, Plymouth, PL4 8AA, United Kingdom (email: {martin.peniak, a.cangelosi}@plymouth.ac.uk).

This work was supported by a EU FP7 Italk and Poeticon++ projects.

increase as well. However, it has been demonstrated that the similarities between various sensorimotor sequences result in competition between the modules that represent them. This leads to a conflict between generalisation and segmentation, since generalisation requires the representation of motor primitives through many similar patterns present in the same module whereas different primitives need to be represented in different modules to achieve a good segmentation of sensorimotor patterns. Because of the conflict that arises when there is an overlap between different sensorimotor sequences, it is not possible to increase the number of motor primitives by simply adding extra low-level modules [13]. The learning of motor primitives (low-level modules) and sequences of these primitives (hi-level modules) need to be explicitly separated through subgoals [14], [12].

Yamashita and Tani [15] were inspired by the latest biological observations of the brain to develop a completely new model of action sequence learning known as Multiple Timescales Recurrent Neural Network (MTRNN). The MTRNN attempts to overcome the generalisation-segmentation problem through the realisation of functional hierarchy that is neither based on the separate modules nor on a structural hierarchy. Hierarchies are rather based on multiple time-scales of neural activities that are responsible for the process of motor skills acquisition and adaptation, as well as perceptual auditory differences between formant transition and syllable level [16], [17], [18], [19], [20].

Neural networks have been used in many different robot motor-control experiments, however, so far the complexity of these neuro-controllers have remained at the similar level. The focus of this paper is to demonstrate that it is possible to scale-up these neuro-robotic controllers with GPUs (Graphics Processing Unit) leading to richer, more realistic and more complex motor control. It is also worth noting that, when these neuro-controllers reach certain sizes, the forward activation will take significant time on standard CPUs, which renders these controllers unsuitable for real-time robot control tasks with typical update time of 50-100ms. The most computationally intesive operations in both traiing and running of neural networks are typically matrix-vector multiplications, which are an ideal match for the SIMT (Single Instruction Multiple Threads) model of a modern GPU processor notoriously famous for outperforming CPUs in parallel computing tasks.

II. Motivation

Around the year 2003, to overcome the energy consumption and heat-dissipation problems of standard PC processors, manufacturers started to produce computers with multiple cores. In the meanwhile, manufacturers have been looking into new technologies that would increase the number of transistors per wafer. However, reducing these dimensions comes at a price since the current leakage becomes a problem.

Since 2003, the production of semiconductors has been divided into multicore and manycore design trajectories. Manycore design aims to increase the processing power by increasing the number of cores in a processor. This number was doubling with each semiconductor process generation starting with dual-core chips and reaching hyper-threaded hexa-core systems. A manycore system is fundamentally different with regards to its design philosophy. While CPUs are optimised for the processing of sequential code and feature sophisticated control logic and large cache memories, the GPU design philosophy emerged from the fast growing video industry where massive numbers of floating point operations are required to render every single frame. As a result, a GPU chip has most of its area dedicated to processing of the floating point operations and features only tiny cache memories.

In 2006, NVidia released GeForce 8800 GPU, which was capable of mapping separate programmable graphics processes to an array of GPUs, which paved the way to first general purpose computing using parallel GPU processors. GPGPU was an intermediate step where graphics card programmers had to use the OpenGL or DirectX API to implement their programs. Using the GPGPU technique many different applications have achieved dramatic speed improvements.

Parallel computing using GPU devices is being increasingly taken up by industry and academies. Many commercial and research applications have migrated from using solely standard CPU processors to a heterogeneous CPU-GPU environments where each architecture does what is best at. Most of these applications achieve tremendous speed-ups in performance [21].

Since quantum computing is still in its infancy and CPUs are approaching the processing limits constrained by the physical laws, we have adopted the heterogeneous CPU-GPU computing paradigm and implemented Aquila 2.0 Cognitive Robotics Architecture (REF). Aquila implements various hi-performance modules that help conducting scientific experiments in the field of cognitive robotics. One of these modules is the above-mentioned MTRNN, which has been optimised for the latest NVIDIA Kepler GPU architecuture. This allowed us to scale-up MTRNN and use it to control all the 53 motors of the iCub humanoid robot (see section III-A) and thus produce more complex and realistic motor control.

III. Method

A. iCub Humanoid Robot Platform

The iCub (www.icub.org) [22] is a small humanoid robot that is approximately 105cm high, weights around 20.3kg and its design was inspired by the embodied cognition hypothesis. This unique robotic platform with 53 degrees of freedom (12 for the legs, 3 for the torso, 32 for the arms and six for the head) was designed by the RobotCub Consortium [23], which involves several European universities and it is now widely used by the iTalk project and few others. The iCub platform design is strictly following open-source philosophy and therefore its hardware design, software as well as documentation are released under general public

license (GPL). Tikhanoff et al. have developed an open-source simulated model of the iCub platform [24], [25]. This simulator has been widely adopted as a functional tool within the developmental robotics community, as it allows researchers to develop, test and evaluate their models and theories without requiring access to a physical robot.

While some of our preliminary experimnets used MTRNN to control all the 53 joints of the iCub, in this experiment we did not need to use legs and therfore only needed to control 41 joints (see table I). The sensorimotor states of the iCub were sampled at 50ms rate and were used for training both the self-organising maps (section III-B) and MTRNN section (III-C). The next section describe the MTRNN model in detail.

body part	degrees of freedom
head	6
left arm	16
right arm	16
torsol	3

TABLE I

DEGREES OF FREEDOM USED.

B. Self Organising Maps for Input Sparse Encoding

The MTRNN system used Self-Organising Maps (SOMs) as means of preserving the topological relations in the multidimensional input space to reduce the possible overlap between various sensorimotor sequences and to aid the learning process.

The self-organising map was trained prior to the MTRNN's BPTT training using a slight variation of the standard SOM unsupervised learning algorithm [26]. The data set consisted of all the sequences used to for the MTRNN training as well as additional sequences, which involved variations to achieve smoother representation of the input space and minimise data loss incurred during the process of vector transformation. Equation (1) shows the description of these vectors where $l(i)$ defines their dimensions.

$$v_i = \{v_{i,1}, v_{i,2}, v_{i,3}, ..., v_{i,l(i)}\} \tag{1}$$

The transformation of a vector to a self-organising map (SOM) is given by equation (2) where $v^{sample} = l(i)$, σ defines the distribution shape of $p_{i,t}$ and N represents the overall size of the self-organising map.

$$p_{i,t} = \frac{exp\left\{-\frac{||v_i - v^{sample}||^2}{\sigma}\right\}}{\sum_{j \in N} exp\left\{-\frac{||v_i - v^{sample}||^2}{\sigma}\right\}} \tag{2}$$

The neural activations on the output layer are assumed to correspond to an activation probability distribution of the self-organising map whose inverse transformation generates multidimensional vector that directly sets the target joint angles of the iCub. Equation (3) describes this transformation where v_i represents the target position for the i^{th} joint index, $y_{j,t}$ is the MTRNN's j^{th} output activity, s_{ij} is the i^{th} index of the vector corresponding to the SOM's node j.

$$v_i = \sum_{j \in N} y_{j,t} s_{ij} \tag{3}$$

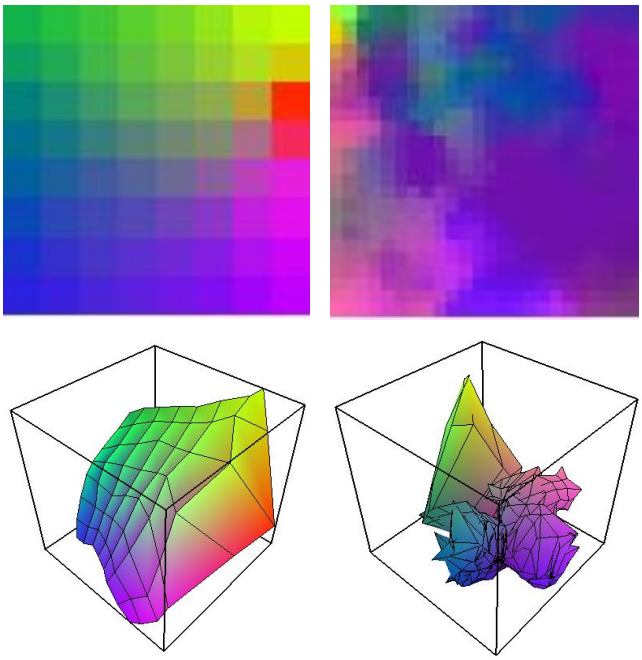

Fig. 1. Trained self-organising map. The picture on the left side shows the map used for encoding the vision (6 joints) and the picture on the right shows map encoding proprioception (35 joints).

C. Online Control

The MTRNN's core is based on a continuous time recurrent neural network characterised by the ability to preserve its internal state and hence exhibit complex dynamics. The system receives sparsely encoded proprioceptive input from the robot (see section III-B), which is used to predict next sensorimotor states and therefore acts as a forward kinematics model (e.g. [27]).

The neural activities were calculated following the classical firing rate model where each neuron's activity is given by the average firing rate of the connected neurons. In addition to this, the MTRNN model implements a leaky integrator and therefore the state of every neuron is not only defined by the current synaptic inputs but also considers its previous activations. The differential equation (4) describes the calculation of neural activities over time where $u_{i,t}$ is the membrane potential, $x_{j,t}$ is the activity of j^{th} neuron, w_{ij} correspond to synaptic connections from the j^{th} to the i^{th} neuron and finally the τ parameter that defines the decay rate of i^{th} neuron.

$$\tau_i u_{i,t} = -u_{i,t} + \sum_j w_{ij} x_{j,t} \tag{4}$$

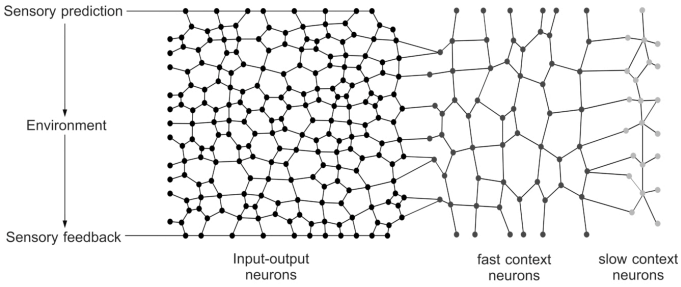

Fig. 2. The system receives proprioceptive information as a multidimensional vector m_t subsequently activating a self-organising map, the activity of which is associated to the network's input. The neural network then predicts the next sensorimotor state m_{t+1} based on its current state and input. At this stage, the neural activations on the output layer are assumed to correspond to the activity of the self-organising map whose inverse transformation generates multidimensional vector that directly sets the target joint angles of the iCub.

The decay rate parameter τ modifies the extent to which the previous activities of the neuron affect its current state. Therefore, when the neurons are set with large τ values their activities will be changing more slowly over time as compared to those neurons set with smaller τ values.

In this experiment, 640 *input-output neurons* were set to $\tau = 2$ while the hidden neurons consisted of two different categories where each had a different time integration constant. The first category comprise of 64 *fast neurons* with $\tau = 5$ and the second of 64 *slow neurons* set to $\tau = 70$. These two categories are attempting to capture the dynamics of complex behavioural patterns by flexible recombination of motor primitives into novel sequences of actions. As described in the introduction, the multiple timescale systems have been suggested as the underlying system that facilitates this behavioural compositionally.

The network is fully connected and hence every neuron is connected to every other neuron including itself. There is one exception where the *slow neurons* are not directly connected to the input-output layer but rather indirectly via the *fast neurons*.

The continuous time integration model of the MTRNN's neurons were defined by the differential equation (4) while the actual membrane potentials are calculated by its numerical approximation defined by equation (8).

$$u_{i,t+1} = \left(1 - \frac{1}{\tau_i}\right) u_{i,t} + \frac{1}{\tau_i}\left[\sum_{j \in N} w_{ij} x_{j,t}\right] \tag{5}$$

The activity of neuron is calculated in two different ways (equation (6)) depending on whether a neuron belongs to the input-ouput ($i \in Z$) or the hidden layer.

$$y_{i,t} = \begin{cases} \frac{exp(u_{i,t})}{\sum_{j \in Z} exp(u_{j,t})} & \text{if } i \in Z \\ f(u_{i,t}) & \text{otherwise} \end{cases} \tag{6}$$

Therefore, the input-output neuron activations are calculated using the Softmax function (the top part of equation (6)) while the hidden neurons use conventional Sigmoid function (equation (7)).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

The Softmax function was used to achieve an activation distribution that is consistent with that of the self-organising map. The system receives proprioceptive information as a multidimensional vector m_t subsequently activating a self-organising map, the activity of which is associated to the network's input. The neural network then predicts the next sensorimotor state m_{t+1} based on its current state and input. At this stage, the neural activations on the output layer are assumed to correspond to the activity of the self-organising map whose inverse transformation generates multidimensional vector that directly sets the target joint angles of the iCub. The iCub then updates the positions of its joints, which are again fed back through the SOM into the MTRNN system as $x_{i,t+1}$. Hidden neurons are simply copied as the recurrent states for the next time step, see equation (8).

$$x_{i,t+1} = \begin{cases} p_{i,t+1} & \text{if } i \in 0 \\ y_{i,t} & \text{otherwise} \end{cases} \tag{8}$$

D. Back Propagation Through Time

The MTRNN needs to be trained via an algorithm that considers its complex dynamics changing through time and for this reason we used the BPTT algorithm as it has been previously demonstrated to be effective with this recursive neural architecture [15].

This learning process is defined by finding the suitable values for the synaptic connections minimising the global error parameter E, which represents the error between the training sequences and those generated by the MTRNN. The error E is calculated using the Kullback-Leibler divergence as described in equation (9) where $y^*_{i,t}$ is the desired activation value of the i^{th} output neuron at the time t and $y_{i,t}$ is its actual output.

$$E = \sum_t \sum_{i \in O} y^*_{i,t} log\left(\frac{y^*_{i,t}}{y_{i,t}}\right) \tag{9}$$

The synaptic connection values are updated according to equation (10) where their optimal levels are approached through minimising their values with respect to $\partial E/\partial w$ that defines the gradient. The learning rate is given by α parameter and n represents the learning iteration step.

$$w_{ij}\,(n+1) = w_{ij}\,(n) - \alpha \frac{\partial E}{\partial w_{ij}} \tag{10}$$

The already mentioned gradient $\partial E/\partial w$ is defined by equation (11) while the recurrence equation (6) is used to recursively calculate $\partial E \partial ui, t$.

$$\frac{\partial E}{\partial w_{ij}} = \sum_t \frac{1}{\tau_i} \frac{\partial E}{\partial u_{i,t}} x_{j,t-1} \tag{11}$$

$$\frac{\partial E}{\partial u_{k,t}} = \begin{cases} y_{i,t+1} - yi^{*}_{i,t+1} + \left(1 - \frac{1}{\tau_i}\right) & \text{if } i \in 0 \\ \sum\limits_{k \in N} \frac{\partial E}{\partial u_{i,t+1}} \left[\delta_{i,k}\left(1 - \frac{1}{\tau_i}\right) + \frac{1}{\tau_k} w_{ki} \int^{'}(u_{i,t})\right] & \text{otherwise} \end{cases} \tag{6}$$

The $\int^{'}()$ is the derivative of the sigmoid function defined by equation (7). The $\delta_{i,k}$ is Kronecker's delta, which is set to 1 when $i = k$ otherwise it is 0.

The initial values of the synaptic connections were randomly generated between -0.025 and 0.025.

IV. Experiments and Results

This experiment was designed to test the scaled-up MTRNN and its ability to learn a complex action while controlling a high number of joints in real-time. The task required the robot to touch an object on a table with either left or right hand depending on the position of the object. If the object is located more on the right side of the table, the robot would touch it with the right hand and vice versa. The object could be positioned anywhere within the rectangular area outlined by the outer circles in fig. 3 below.

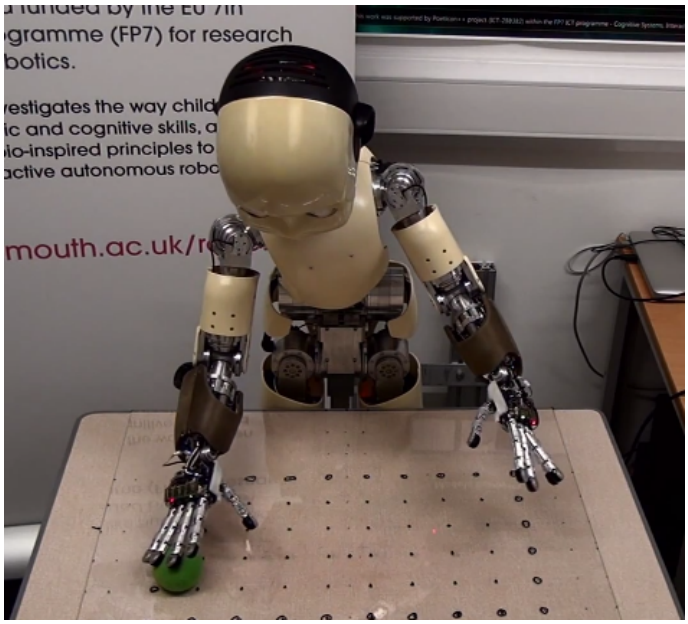

Fig. 3. Experimental setup

The Sequence Recorder module of Aquila was used to record the sensorimotor patterns while the experimenter was guiding the robot by holding its arms and performing the action for each object position defined by the inner 32 circles separed by 5cm distance (see fig. 3). Each recording lasted 5 second and the encoder values of 41 joints were sampled at 50ms interval, producing the total of 3200 sensorimotor states used for the training of self-orgnainsing maps and MTRNN.

During this data collection stage, the Aquila's tracker module was used to keep the robot's head and eyes centered on the object regardless of its position. Different object positions would therefore yield different encoder values of the 6 joints in the head and eyes. Since this information was passed as the input, the MTRNN was able to distinguish between different object positions and perform the action in the correct way.

A total number of 20 trials were conduced where each could run for the maximum of 10000 iterations. Each training trial was initialised with different initial seed used for generating neural network weights. At the end of the training, the neural network from each trial was tested on the robot, which needed to be able to touch the object at the correct position and with the correct arm.[1]

When the neural network error was less than approximatelly 0.00001, the robot was able to successfully execute the action whatever the object position was. Our experimental results show (see table II) that 15 our of 20 trials were successful and resulted in capable neuro-controllers.

run	error
1	0.000009
2	0.000108
3	0.000033
4	0.000019
5	0.000006
6	0.000009
7	0.000007
8	0.000008
9	0.000007
10	0.000008
11	0.000007
12	0.000007
13	0.000007
14	0.000009
15	0.000042
16	0.000007
17	0.00001
18	0.000006
19	0.000009
20	0.000035

TABLE II
Training Results

V. Conclusions

Neural networks have been used in many different robot motor-control experiments, however, so far the complexity of these neuro-controllers have remained at the similar level. In this paper, we have demonstrated the it is feasable to scale-up neural networks using GPUs and thus develop complex nero-controllers able to controll a high number of joints leading to more realistic action execution. Our preliminary experiments also suggest that it is possible to train high number of different actions using a single neural network with higher number of neurons.

[1]This video shows one of the trained neural networks controlling iCub humanoid robot: http://www.youtube.com/watch?v=PtaPaEjkMJ0

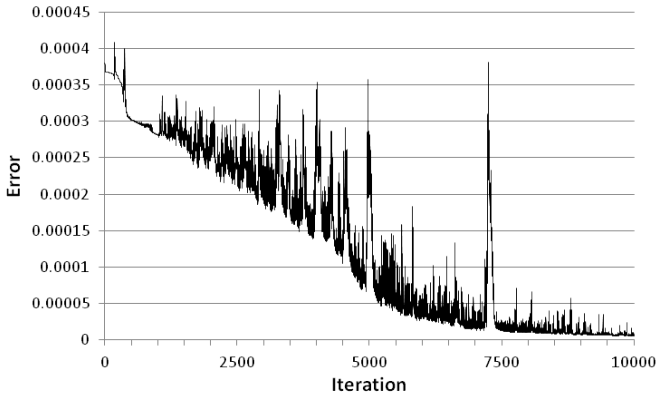

Fig. 4. Training errors of the best trial.

REFERENCES

[1] M. Jeannerod, *The Cognitive Neuroscience of Action*. Cambrige, MA and Oxford UK, Blackwell Publishers Inc, 1997.

[2] G. Rizzolatti and G. Luppino, "The cortical motor system," *Neuron*, vol. 31, pp. 889–901, 2001.

[3] L. Fogassi, P. F. Ferrari, B. Gesierich, S. Rozzi, F. Chersi, and G. Rizzolatti, "Parietal lobe: From action organization to intention understanding," *Science*, vol. 308, no. 4, pp. 662–667, 1996.

[4] A. Graybiel and W. Johnson, "A comparison of the symptomatology experienced by healthy persons and subjects with loss of labyrinthine function when exposed to unusual patterns of centripetal force in a counter-rotating room," *Ann Otol Rhinol Laryngol*, vol. 72, no. 2, pp. 357–374, 1963.

[5] J. R. Lackner and P. DiZio, "Adaptation in a rotating artificial gravity environment," *Brain Research Reviews*, vol. 28, pp. 194–202, 1998.

[6] K. Sakai, K. Kitaguchi, and O. Hikosaka, "Chunking during human visuomotor sequence learning," *Experimental Brain Research*, vol. 152, pp. 229–242, 2003.

[7] K. A. Thoroughman and R. Shadmehr, "Learning of action through adaptive combination of motor primitives," *Science*, vol. 407, pp. 742–747, 2000.

[8] A. d'Avella, A. Portone, L. Fernandez, and F. Lacquaniti, "Control of fast-reaching movements by muscle synergy combinations," *Neuroscience*, vol. 26, no. 30, pp. 7791–7810, 2006.

[9] M. S. Graziano, C. S. Taylor, T. Moore, and D. F. Cooke, "The cortical control of movement revisited," *Neuron*, vol. 36, pp. 349–362, 2002.

[10] S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi, "Convergent force fields organized in the frog's spinal cord," *Neuroscience*, vol. 13, pp. 467–491, 1993.

[11] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, pp. 1317–1329, 1998.

[12] J. Tani and S. Nolfi, "Learning to perceive the world as articulated: an approach for hierarchical learning in sensory–motor systems," *Neural Networks*, pp. 1131–1141, 1999.

[13] J. Tani, R. Nishimoto, J. Namikawa, and M. Ito, "Codevelopmental learning between human and humanoid robot using a dynamic neural-network model," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 38, pp. 43–59, 2008.

[14] J. Tani, R. Nishimoto, and R. Paine, "Achieving "organic compositionality" through self-organization: reviews on brain-inspired robotics experiments," *Neural Networks*, vol. 21, pp. 584–603, 2008.

[15] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment," *PLoS Computational Biology*, vol. 4, no. 11, 2008.

[16] K. Newell, Y. Liu, and G. Mayer-Kress, "Time scales in motor learning and development," *Physical Review*, vol. 108, pp. 57–82, 2001.

[17] R. Huys, A. Daffertshofer, and P. J. Beek, "Multiple time scales and multiform dynamics in learning to juggle," *Motor Control*, vol. 8, pp. 188–212, 2004.

[18] F. Varela, J. P. Lachaux, E. Rodriguez, and J. Martinerie, "The brainweb: phase synchronization and large-scale integration," *Nature Reviews Neuroscience*, vol. 2, pp. 229–239, 2001.

[19] C. J. Honey, R. Kotter, M. Breakspear, and O. Sporns, "Network structure of cerebral cortex shapes functional connectivity on multiple time scales," *Proceedings of the National Academy of Sciences*, vol. 368, pp. 10 140–10 245, 2007.

[20] D. Poeppel, W. J. Idsardi, and V. van Wassenhove, "Speech perception at the interface of neurobiology and linguistics," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 368, pp. 1071–1086, 2008.

[21] NVidia, "Cuda community showcase," December 2010. [Online]. Available: www.nvidia.com/object/cuda_apps_flash_new.html

[22] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, "The icub humanoid robot: An open-systems platform for research in cognitive development," *Neural Networks*, vol. 23, no. 8-9, pp. 1125–1134, 2010.

[23] G. Metta, D. Vernon, L. Natale, F. Nori, and G. Sandini, "The icub humanoid robot: an open platform for research in embodied cognition," in *IEEE Workshop on Performance Metrics for Intelligent Systems*, 2008.

[24] V. Tikhanoff, P. Fitzpatrick, F. Nori, L. Natale, G. Metta, and A. Cangelosi, "The icub humanoid robot simulator," in *International Conference on Intel ligent RObots and Systems IROS*, Nice, France, 2008.

[25] V. Tikhanoff, A. Cangelosi, and G. Metta, "Integration of speech and action in humanoid robots: icub simulation experiments," *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 1, pp. 17–29, 2011.

[26] T. Kohonen, *Self-Organizing Maps*. Springer, 1996.

[27] D. M. Wolpert, Z. Ghahramani, and M. I. Jordan, "An internal model for sensorimotor integration," *Science*, vol. 269, pp. 1880–1882, 1995.