Rewarded multi-modal neuronal self-organization: Example of the arm reaching movement

Olivier Ménard, Hervé Frezza-Buet

Rewarded multi-modal neuronal self-organization: Example of the arm reaching movement

Olivier Ménard
Loria[1], Supélec[2]
[1] Bât Loria, Campus Scientifique,
BP239, F-54506 Vandœuvre-lès-Nancy
Email: Olivier.Menard@supelec.fr

Hervé Frezza-Buet
Supélec[2]
[2] 2, rue Edouard Belin
F-57070 Metz Cedex
Email: Herve.Frezza-Buet@supelec.fr

***Abstract*— This paper presents a computational model of cortical organization, enhancing the fact that self-organization in connected cortical modules have to be joint. This architecture is validated on the learning of the control of a robotic arm device, through sensori-motor integration. In order to get rid of the use of an inverse model of the arm, the self-organization of the whole network is conditioned by a reward signal. The point of the model is that this reward signal is given at the motor level only, but its effects are spread to the whole architecture by a side effect of the coherence keeping function in the cortical modules. First experiments on this approach are discussed.**

I. Introduction

Current understanding of the primate brain organization leads to consider the nervous system as a set of subsystems, highly connected. They provide complementary and collaborative information processing skills [1]. Among these numerous subsystems, several ones can be viewed as providing some *generic* processing power, that can be dedicated for the actual relevant processings needed by the primate. For example, the hippocampus is concerned with the ability of storing recent facts, keeping an integrated representation of the perceptual and emotional qualities of the stored events, whatever the modalities these events imply. Let us also consider the function of the cerebellum as related to the ability of learning complex automatic relations from examples of any kind, the role of the basal ganglia as providing decisional ability, for motor processings for example, based on a reward-related competition mechanism [2]. The example that mostly concerns the work presented in this paper is the cerebral cortex, that is a surface where information is scattered and organized, according to its modality. The function of the cortex can be viewed as keeping information coherent at different levels of abstraction [3], [4]. This is a generic process since this coherence is maintained for many kinds of processings. This stands at perceptive level (posterior cortex), as illustrated by the LAMINART model [5] where maintaining image coherence at different spatial scales helps to remove the noise in input pictures. This also stands at the behavioral level (prefrontal cortex), since keeping behavior coherent requires the shutting down of distractive action schemes. We refer to [6] for an excellent overview concerning this last point, and its relation to human articulated speech and reasoning capabilities.

All these functional views are of course very rough interpretations of the complexity of the brain, but this functional level of understanding may help us to consider the brain systems as implementing different paradigms of information processing. This is what the model presented in this paper illustrates concerning the cortex. This work is actually an attempt to use paradigms inspired from biological models of cortical processing in order to provide an effective computational model of sensorimotor integration. This is applied to an elementary arm-reaching robotic task, so that the model can be discussed in terms of performance and feasibility.

II. Cortical properties addressed by the model

One main challenge when addressing a cortical model from a computational point of view is to solve the so called *binding* problem, as two properties of cortical processing mays sound opposite. The first one is the fact that information is scattered over the cortical surface. That means for example that for an event, its visual component is managed by the visual cortex, whereas its auditory aspect is managed by the auditory cortex, and its consequences are managed at the level of the prefrontal cortex, and so on. Moreover, even for the very visual aspect of that event, textures, contrast detections, colors and motions are managed by different visual cortical areas. The role of such a scattering for generalization of information has been shown in some computational model in the past [7]. The other property of cortical computation is that information is kept coherent, in spite of its previously mentioned scattered nature. The model presented here shows a way to learn modal components of the arm-reaching problem in different modules, whereas global coordination of the modules is preserved by a strictly local mechanism, without needing any supervisory algorithmic module.

Another point with cortical modeling is the topographic organization of neurons in cortical modules. That means that neighboring neurons on the cortical surface are concerned by slightly different information. A well-known example is the selectivity to contrast orientation in visual cortex, where neighboring neurons detect contrasts at quite the same place in the retina with different, but close, orientation [8]. From a computational point of view, this has been studied from decades at both architectural and learning point of view [9],

leading after simplification to the famous Self Organizing Maps (SOMs) by Kohonen [10]. Those models involve local competition mechanisms, allowing only winning neurons to learn. This competition mechanism as itself is grounded on sharp local excitatory connections and longer range, but weaker, inhibitory connections. Its stability has been first studied in the framework of the Continuum Neural Field Theory (CNFT) by Amari [11], and has been generalized to bi-dimensional neural fields by Taylor [12]. Some recent RFLIS-SON model [13] has addressed the feasibility of learning the lateral excitatory and inhibitory pattern, but, in spite of good fitting to biological data, it suffers, from a computational point of view, from needing specific initialization.

Our model addresses the self-organization property of the cortex in a similar way, but it extends self-organization to many modules, as already done for the arm control [14]. The connection between the modules are partial, organized into stripes of axons as detailed in section IV-C. That means that neighboring neurons receive slightly different information, which forbids the use of Kohonen SOMs and their winner-takes-all mechanism. That is the reason why we implement a CNFT-like local competition, but the model doesn't address yet the problem of learning the involved lateral connection weights, keeping constant the classical on-center off-surround pattern.

As our approach concerns computer science, we insist on the following properties of the model, that are not always fulfilled by the models mentioned above, since their very purpose is rather to explain biological observations. First property is that no constraint is required for initialization, all the experiments are made from initial random weight and activity values. The second property is the stationarity of all the model parameters. This means that no progressive decrease of neural plasticity is performed, as often required by algorithms such as the SOM one. Moreover, there is no specific learning stage, learning being performed on-line, allowing the system to work better and better as long as it is used. Last, the entire computation is made strictly local, a neuron accessing only the state of connected neurons to compute its activity. This forbids any supervision of the algorithm and makes the model intrinsically parallel [15].

As already shown in [16], the model presented here allows to maintain coherence in partially connected modules. Whereas each module is concerned with one specific modality of the arm reaching task problem (articulatory position, articulation motion, hand position in Cartesian space), learning is performed so that the organization of the different modules overcomes the partial connectivity. That means that, after learning, neurons in different modules, that have specialized to managed some information, stand at places in each module that are actually connected if the corresponding information are related.

The genericity of the model has also been shown, since it has been used to propose a mechanism for the organization to word-related cortical areas according to body representation, thus bringing semantics to the acoustical features of the words [17]. Nevertheless, there is still a difficulty that should be addressed by our model for it to be usable for real robotic application. This difficulty is that the model has to be applied in cases where both direct and inverse model of the robotic device are unknown. This led us to introduce reward-related mechanism in the model architecture, which is the central point of the present paper.

III. The need of reward

The arm-reaching task can be viewed as a servo-controlled process, and thus can be thought as quite far from goal-oriented cognitive decision tasks. These latter ones have been modeled in the framework of Markov Decision Processes (MDPs) [18], where a formal relation between action selection and reward optimization has been set up. The point of the discussion in this paper is that unrewarded usual unsupervised learning, as the one by SOMs and related techniques, cannot easily lead to the learning of sensori-motor skills. Let us illustrate this difficulty by the example of the arm reaching task, keeping in mind that this may also stands for any servo-control cortical learnings.

Learning to control the hand position through muscles activation means learning to transform a target in the visual space (desired hand position) into a muscle activity. As shown by the biological model by Baraduc and Guigon [19], this learning consists in finding the variation of muscles position to move the hand toward the desired position, knowing current angles of articulation (proprioception). One could think that a good way to learn this transformation is to generate from current arm posture random muscle contractions, observing the effect on the hand position in space, and learn this by a self-organizing process. The problem of such an approach is that the distribution of the hand positions generated by random actions is not uniform in the hand position visual space. As usual unsupervised techniques are sensitive to distribution density, the cases that are learned in that context are concentrated on some positions in the visual space. For the control of the hand, the network has to be able to deal with any position of the hand in the visual space. As some (and often many) of these useful positions are made seldom by the random action process, they are not learned at all. Since models as the one by Baraduc and Guigon rather focus on biological data fitting, authors can afford using the inverse model of the arm to feed the model with examples that are well distributed in the visual space rather than examples chosen randomly in the motor space. This is also what has been done in [16] that was focusing rather on joint organization properties of our model than on this very point. Nevertheless, in our computer science context, as we aim to address devices for which direct and inverse models are unknown, it is therefore necessary to find another way of having well distributed examples.

The idea that is proposed in this paper is to still present examples to the model that are generated from random actions, but to bias the learning process so that it favors "well working" examples. That means that we define a target to be reached by the model, let the model generate stochastic actions, and

learn with more intensity the ones (few at the beginning of the learning process) that actually lead to the reaching of the target. Whether the target has been reached or not by the hand is provided to the model by a reward signal, which makes the theory of MDPs enter our self organizing process. Nevertheless, having an accurate model of the reward signals provided by the basal ganglia is out of the scope of the present work, and this part will be over-simplified here. Using this simplification, we can focus on the goal to make only one small piece of learning in the model be reward sensitive. This small part stands at motor modules level only, and the spreading of the reward sensitivity to all the modules is a side effect of the model's ability to maintain coherence between the modules. Thus, even if the learning rules in the different modules are self-organizing rules, without any reinforcement learning properties, the computation all over the architecture takes reward into account, and therefore overcomes the problem of not having an inverse model of the robotic device at disposal for generating well distributed examples.

IV. `bijama`, A GENERIC CORTICAL DESIGN

The very purpose of the `bijama` model (stands for `b`iologically `i`nspired `j`oint `a`ssociative `ma`ps) is to provide a generic framework for the design of computational models inspired from some cortical features.

A. *The* `bijama` *maps*

The cortical computation in the model is divided into different modules, called `bijama` maps. These modules are 2D sets of computational units, that recalls the fact that the cortex is a sheet of elementary neural circuits, the cortical columns [3].

B. *The* `bijama` *units*

The units in the model have to compute an activation from several kinds of information. This activation is noted $A^\star$, and is obtained from another A^g activity, detailed further, also computed by the unit. Actually, $A^\star$ is the result of a competition from the A^gs of neighboring units in the module. As it is inspired from the CNFT [12], this competition is the result of an on-center off-surround influence of the units in a module. The result of that competition is that some compact patches of $A^\star$ activity raise at the places where A^g activity is locally maximal in the module. This pattern of $A^\star$ activities over the the module is called bubbles of activities, and correspond to some stable state of the on-center off-surround relaxation mechanism.

Let us now present the computation of the A^g activity, from which competition is made to obtain resulting $A^\star$ bubbles (see bottom left part of figure 1). The A^g activity is a merging of two kinds of components.

The first kind is a so-called thalamic component A^θ, that correspond to an external input given to the unit. The word thalamic recalls the fact that some extra-cortical inputs are provided to the cortex by the thalamus, but the model is not accurate on that point. The A^θ activity is the result of the matching of an input with an inner prototype of the unit, as in the SOM algorithm. This A^θ activity is strong if the currently presented input to the unit is close to the prototype value stored by that unit.

The second component that is merged with A^θ to form A^g is the cortical activity A^c. This activity is also a combination of several computations. Each of these computations corresponds to some connection with units that are all from the same remote module. In figure 1, the highlighted unit in module C, for example, is connected to three other modules (two are shown, one other is suggested), like all other units in module C. Many units in each module are connected to this unit. A computation is made for each module with the connected units belonging to them. In the example on figure 1, for a remote module i, a computation A^c_i is made as the weighted sum $A^\star$ activities of connected units in module i. Then, A^c_A, A^c_B and A^c_D are combined to form global cortical A^c for a unit in C.

To sum up, for a unit (see bottom left part of figure 1), a computation of an activity A^c_i is done for each remote module i the unit is connected to. Then, these A^c_i are combined, implementing a logical AND function, to form A^c. This A^c value is combined the same way to the matching result A^θ of the input with inner prototype to form A^g. Last, A^g is the basis of a competition among neighboring units in the module, and leads to the raising of compact bubbles of $A^\star$ activities. Even if they are described here sequentially, all these processes run simultaneously in the model (see [16] for details).

Last, let us mention that some modules may have units that receive no external input and compute no A^θ activity. This is the case for associative modules that deal with no inputs but link some modalities together. Moreover, the number of cortical activities A^c_i used to form A^c depend on the number of other modules the module is connected to.

C. *The modular stripes*

One main feature of the `bijama` model is that it allows the definition of *several* modules (or maps). Each one is a self-organizing process, but the connections between the modules make these processes coupled. This coupling will be discussed in next paragraph, once the connectivity of the model is presented.

Some local classical on-center off-surround connectivity stands among units in a module, that allows the A^g activities to compete to form bubbles of $A^\star$ activity, as already discussed. What is more specific to the `bijama` model is the connection pattern between two connected modules. Let us consider connections from module A to module C on figure 1. All units in module A send *parallel* axons fibers toward C. These axons cross the module C, providing the units they "cross" an access to the unit in A an axon belongs to. To sum up (see right part of figure 1), a given unit in C receives inputs from three stripes of units, respectively from A, B and D. This kind of connectivity is inspired from modular stripes described in [3].

Even if this is not mandatory with the `bijama` model, connections are set reciprocal in our experiments. That means

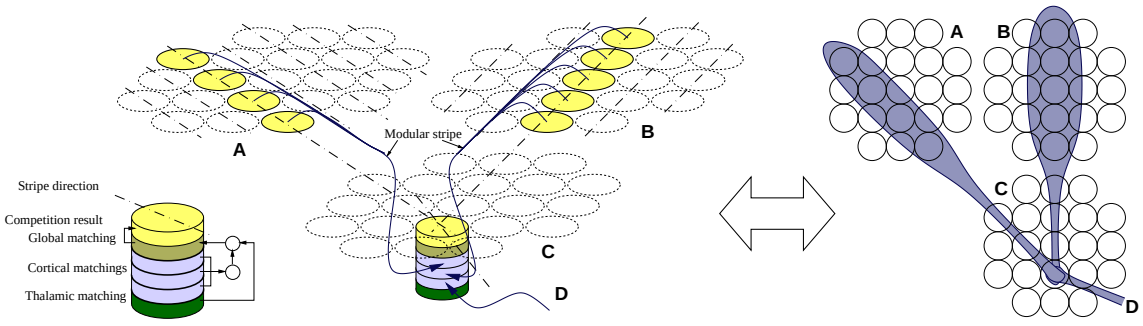

Fig. 1. The bijama architecture. Modular stripes allow partial connectivity between modules. A unit has to merge the result of a matching with external input A^{θ} (thalamic input) and a combination of all cortical inputs A^{c}_{A}, A^{c}_{B} and A^{c}_{D}, in order to form a global A^{g} activity. A^{g} is the basis of a competition to form bubbles of $A^{\star}$ in each modules.

that when a module sends fibers to another one, this latter one also sends fibers back in the same direction. This allows the setting of resonant loops, as discussed in next section.

Last, as axon fibers may have any direction, it is more suitable not to favor one direction in a map. That's why the maps are made round-shaped, so that they are similar regards to any direction (see figure 1)

D. Coherent learning and joint self-organizing

In the model, learning is done continuously, but it is modulated by the $A^{\star}$ activity of the unit. That means that only locally winning units learn. This learning modifies the inner input prototype used to compute A^{θ}, as in the SOM algorithm. The point in the bijama model is that this learning also modifies the weights of the connections coming from cortical stripes. Thus, the unit learns to be preferentially activated by a specific pattern of $A^{\star}$ activities that stand in a stripe in each remote module it is connected to.

As connections between modules are reciprocal stripes, stable states of activation are states where bubbles of $A^{\star}$ activities stand in each modules at places that are actually connected together. This constraint forces the learning process to be effective only at connected places. The self-organization in each module is then biased so that connected places learn simultaneously. As shown in [17] for a word organization, the topographic organization of features, as in SOMs, is preserved in the module, but the organization that is found also allows related features in different modules to be actually connected.

The effect of such a coherence learning is to overcome the lack of connections, since stripe connectivity is much sparser that a naive all-to-all connectivity. When the modules in the network are stimulated by some external multi-modal input (each involved modality is managed by one module), a distribution of A^{θ} activations is set according to the matching with inner prototypes. Then, the bubbles of $A^{\star}$ try to make the best compromise between these inputs and the fact that they should be coherent. Coherence mean that they should correspond to units that are actually connected, since connected units are the one that are usually activated simultaneously. A resonant process, through reciprocal stripes, drives a relaxation of the network, stabilizing bubbles at places where both fitting to current external state (A^{θ}) and coherence of that state according to previous experiments ($A^{\star}$) are fulfilled.

V. Using reward-dependant motor learning for the control of a robotic arm

The arm reaching problem consists in moving the hand toward a target, and it is visually guided when the video module sees both the hand and the target. Here, two sensory inputs have to be combined in order to make the proper action. One is the visual demand, i.e. the target's position in relation to the hand's position. The other one is the arm's current posture, and is related to the proprioceptive sensory modality. We use two maps, one for each of these modalities, and another one for the arm motor modality. A supplementary associative map links all of these together (cf. fig. 2).

In addition to these elements, two additional features are included, in order to incorporate reward into our model. One is the *spinal* map. This map, connected to the motor map, is where the actual motor actions the model performs are chosen. Each unit in this map corresponds to a specific arm movement. This is hard-wired and not learned. The other additional feature is a unit, called the *basal* unit. That unit takes as inputs all cortical $A^{\star}$ activities, and outputs to all units in the spinal map. Its role is to predict the reward that the model will get when it takes an action.

In order for the model to learn to perform accurate movements, we reward it when the actual arm movement has got the hand closer to the target. This is done by implementing a reinforcement learning mechanism that interacts with the motor part of the model. In our previous work [16], we had not used such a mechanism, and it was required to actually impose on the model to perform only the correct movement even before it had learned. As already said, this was done by using an inverse model of the arm to generate well-distributed examples. This means that the model learns to perform an action that a simpler algorithm (the inverse model) already performs. With reinforcement learning, it can learn something that we don't know how to perform otherwise: We only need to evaluate the model's performance to drive a correct self-organization.

At each time step, the basal unit estimates a prospective reward value from the state of the system, i.e. from the cortical

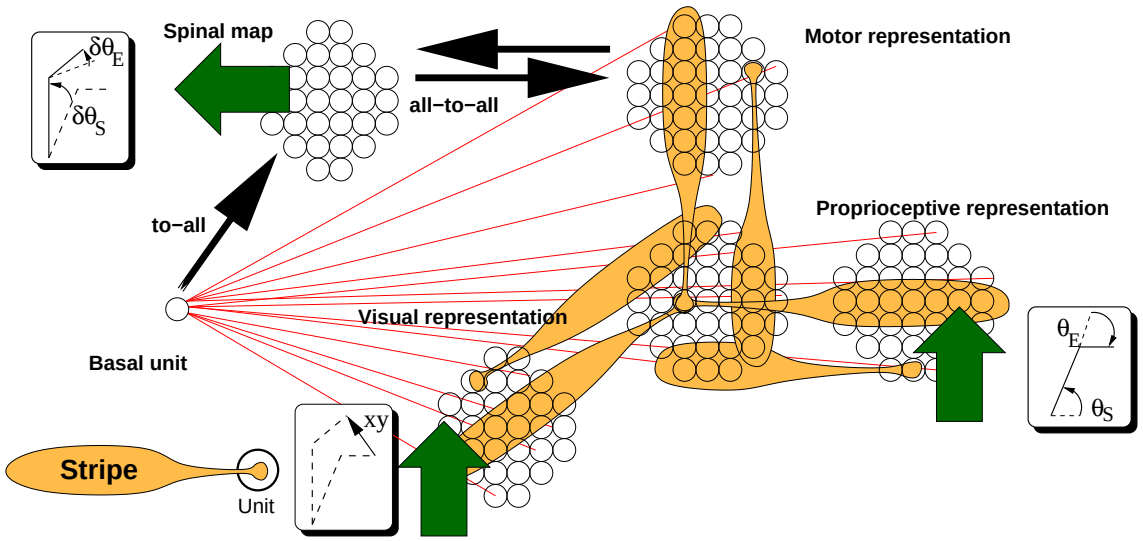

Fig. 2. Our model's architecture. Two input maps, one for visual input and one for proprioceptive input, are linked, together with a motor map, to an associative map. These links are actually modular stripes. The motor map is the only cortical map directly linked to the reward system: the spinal map is completely linked to the motor map. The basal unit takes its input in all cortical maps, and its output is used to modulate A^g activity in the spinal map. The arm posture is represented by the shoulder and elbow angles θ_S and θ_E, the visual demand is represented by a (x, y) Cartesian vector and the motor action is a change of the arm angles $\delta\theta_S$ and $\delta\theta_E$

unit's $A^\star$ activities. Whenever the system actually performs a movement, the real reward is computed from the actual arm movement. The basal unit then learns, modifying the weights of the connections coming from the cortical units, so that the prospective reward gets closer to the actual reward. This corresponds to the policy evaluation stage of usual MDP systems [18].

The prospective reward predicted by the basal unit is used to modulate the choice of the action the system performs. That action is computed inside of the spinal map. As previously explained, all possible motor actions are a priori represented in this map. Its main input is an all-to-all connectivity with the cortical motor map, and not cortical stripe connectivity (cf. fig. 2). This ensures that the dynamic organization of the cortical motor map remains completely independent from the fixed organization of the spinal map.

The spinal map computes its $A^\star$ activity from the cortical input as other `bijama` units do, except that a random activity is added to the A^g activity, which corresponds here to the cortical input. This random activity diminishes as the predicted reward grows. Thus, the system will exploit the situations in which it performs well, and explore other possibilities when its fails to perform promising actions. This ensures that the model will perform better as it learns, since it will only change its behavior when it is incorrect. It corresponds to the policy improvement stage of MDP systems [18].

While the learning rules in the cortical maps are unaffected by the introduction of the reward system, the whole resulting learning process itself is nevertheless reward-related. This is a direct consequence of the resonance that occurs within the system, since the activity bubbles become unstable in the spinal map when the predicted reward is low, due to the random input. The bubbles of activity in the motor map then become unstable too. This instability propagates through the cortical stripe connections in the model. Thus, learning is only efficient when the predicted reward is high. This ensures that the model, as a whole, doesn't learn to perform inappropriate actions.

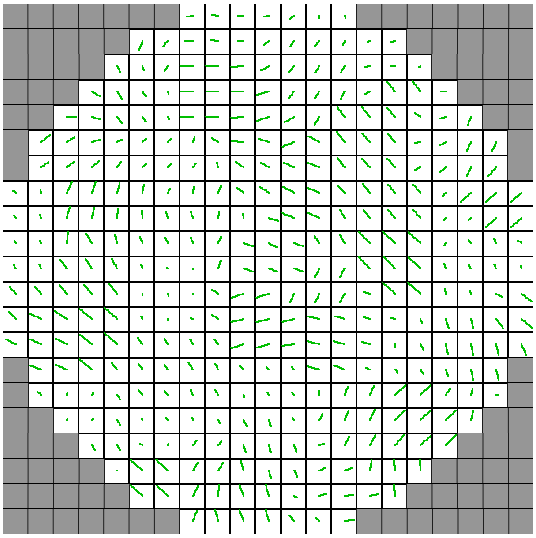

Fig. 3. Thalamic prototypes (x, y) after organization of the visual map.

VI. Experimental Results

When the model has learned, all of the cortical modules self-organize (cf. fig. 3). Their organization is the one that allows coherence to occur as the model processes its different inputs, while remaining continuous when possible, so that close units in the map have close prototypes. Discontinuities

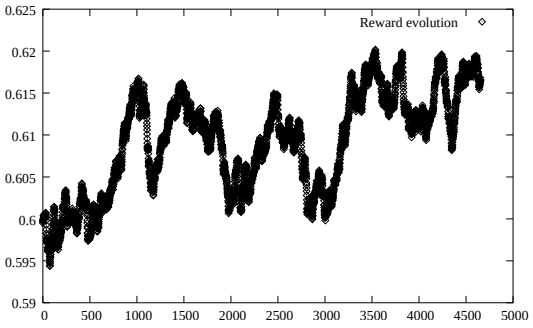

Fig. 4. Reward evolution: Reward is low-pass filtered and a time unit corresponds to 100 time steps

are due to the constraints imposed by the other maps via the stripe connections.

The model computes a reward each time it performs an action. That reward is a measure (in $[0, 1]$) of how close the final arm position (after the movement) is to the target's position. It increases (cf. fig. 4) as the model learns what actions provide the best reward.

VII. Discussion and perspectives

An extension of recent `bijama` cortical model is presented in this paper, adding this model the ability to take a reward signal into account in order to bias the organization of cortical maps. Actuators are introduced in the model by a so called "spinal" map, where neural activation directly corresponds to action through an a priori hard-wired relation. What differs in the work presented here from previous application of the `bijama` model [16], [17] is the fact that the motor cortical map has to learn the mapping of its neurons to the neurons in that spinal map. This degree of freedom, i.e. the fact that many mappings are allowed before the learning process actually converges, is crucial for reward influence, since the learning of this mapping is biased by the quality of the action proposed by the system. This allows the system to produce an activity in the motor map, and to adjust the mapping to actual action so that it improves the command. The emerging behavior of the system is then to learn to *build* a command in the motor map, at a place in that map that is actually connected to the preconditions of that command, coming from other maps through the associative one.

From the early experimental validation presented here, using resonance at the cortical levels seems a general paradigm, as already shown in previous works, but also at the level of the relations between the cortical motor maps and their "peripherals", as the spinal module presented here. It allows in our case to address reinforcement learning without making all the modules reward related, since all the resonant loops between the modules spread consistency all over the network. The model thus illustrates that computational skills can be separated in different modules: coherence preservation is the role of the cortical modules and reward-related learning is kept located between cortical motor module and the "spinal" module. Nevertheless, the point is that this separation doesn't prevent the whole system from getting benefits from both these abilities.

Future extension of this work is a better modeling of the reward system by setting up a more extensive model of the basal ganglia system. Tests of the model presented here on the real robotic arm in our laboratory are currently at work.

References

[1] K. Doya, "What are the computations in the cerebellum, the basal ganglia, and the cerebral cortex," *Neural Networks*, vol. 12, pp. 961–974, 1999.

[2] J. W. Mink, "The basal ganglia: Focused selection and inhibition of competing motor programs," vol. 50, pp. 381–425, 1996.

[3] Y. Burnod, *An adaptive neural network : the cerebral cortex*. Masson, 1989.

[4] D. H. Ballard, "Cortical connections and parallel processing : Structure and function," vol. 9, pp. 67–129, 1986.

[5] S. Grossberg, E. Mingolla, and J. Williamson, "Synthetic aperture radar processing by a multiple scale neural system for boundary and surface representation," vol. 8, pp. 1005–1028, 1995.

[6] J. M. Fuster, *The Prefrontal Cortex : Anatomy, Physiology, and Neuropsychology of the Frontal Lobe*. Lippincott Williams and Wilkins Publishers, 1997.

[7] H. Frezza-Buet and F. Alexandre, "From a biological to a computational model for the autonomous behavior of an animat," *Information Sciences*, vol. 144, no. 1-4, pp. 1–43, july 2002.

[8] D. H. Hubel and T. N. Wiesel, "Functional architecture of macaque monkey visual cortex," *Ferrier Lecture Proc. Roy. Soc. London*, pp. 1–59, 1977.

[9] D. Willshaw and C. von der Malsburg, "How patterned neural connections can be set up by self-organization," *Proceedings of the Royal Society of London*, vol. B 194, pp. 431–445, 1976.

[10] T. Kohonen, *Self-Organization and Associative Memory*, ser. Springer Series in Information Sciences. Springer-Verlag, 1989, vol. 8.

[11] S.-I. Amari, "Dynamical study of formation of cortical maps," *Biological Cybernetics*, vol. 27, pp. 77–87, 1977.

[12] J. G. Taylor, "Neural networks for consciousness," vol. 10, no. 7, pp. 1207–1225, 1997.

[13] R. Miikkulainen, J. A.Bednar, T. Choe, and J. Sirosh, "Self-organization, plasticity, and low-level visual phenomena in a laterally connected map model of the primary visual cortex," *Psychology of Learning and Motivation*, 1996.

[14] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps; An Introduction*. Addison-Wesley Longman Publishing Co., 1992.

[15] O. M´enard, S. Vialle, and H. Frezza-Buet, "Making cortically-inspired sensorimotor control realistic for robotics: Design of an extended parallel cellular program," in *IEEE Advances in Intelligent Systems - Theory and Applications*, 2004.

[16] O. M´enard and H. Frezza-Buet, "Multi-map self-organization for sensorimotor learning: a cortical approach," in *IEEE International Joint Conference on Neural Networks*, 2003.

[17] O. M´enard, H. Frezza-Buet, and F. Alexandre, "Multi-criteria self-organization: Example of motor-dependant phonetic representation for a multi-modal robot," in *AI workshop on Neurobotics, 27th Conference on Artificial Intelligence*, 2004.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*. MIT Press, 1998.

[19] P. Baraduc and E. Guigon, "Population computation of vectorial transformations," *Neural Comput*, vol. 14, no. 4, pp. 845–871, 2002.

Acknowledgment

The authors would like to thank the Lorraine region, the Robea program of the CNRS and the European MirrorBot project for their support.