Multiple Object Manipulation: is structural modularity necessary? A study of the MOSAIC and CARMA models

Stéphane Lallée (stephane.lallee@inserm.fr)
INSERM U846, Stem-Cell and Brain Research Institute
Bron, France

Julien Diard and Stéphane Rousset
Laboratoire de Psychologie et NeuroCognition CNRS-UPMF
Grenoble, France

Abstract

A model that tackles the Multiple Object Manipulation task computationally solves a higly complex cognitive task. It needs to learn how to identify and predict the dynamics of various physical objects in different contexts in order to manipulate them. MOSAIC is a model based on the modularity hypothesis: it relies on multiple controllers, one for each object. In this paper we question this modularity characteristic. More precisely, we show that the MOSAIC convergence during learning is quite sensitive to parameter values. To solve this issue, we define another model (CARMA) which tackles the manipulation problem with a single controller. We provide experimental and theoretical evidence that tend to indicate that non-modularity is the most natural hypothesis.

Keywords: motor control; MOSAIC; CARMA; modularity; internal representation; neural network.

Introduction

There is, in the world, an infinity of objects with different physical behaviors. Despite this variability, humans can manipulate them with ease, from light origamis to heavy cups. For a given goal position, how do they select the correct force to apply? How are they able to accurately predict the displacements resulting from the applied forces? These two questions are central in object manipulation: control and prediction, respectively. If the physical characteristics of objects and their identity are known, or if there is a single object, this problem is easy to model and solve. Indeed, the dynamics of physical bodies are well described by Newton's equations. Given the starting position, and the applied forces, it is straightforward to compute the resulting trajectory.

The problem becomes much more difficult if the objects are numerous, and of unknown physical parameters. We call this the Multiple Object Manipulation task (MOM).

It is thought that natural cognitive systems are able to solve this problem because they are capable of good predictions in uncertain and unstable environments. Modeling this ability can provide insights and a better understanding of the possible brain structures involved in the process (Kawato, 2008). This has lead Gomi and Kawato to propose the MOSAIC model (MOdular Selection And Identification for Control) (Gomi & Kawato, 1993). This model solves both problems of object identification and object control simultaneously. The key feature of MOSAIC is that it uses neural networks in a modular way. In other words, the system has multiple distinct neural controllers, one for each object. It is able to choose which controller to use in order to manipulate an object, even without knowing the object identity explicitly.

The structural matching between object and controller, in MOSAIC, is a very strong hypothesis, that we question in this paper. To do so, we developed another model, CARMA (Centralized Architecture for Recognition and MAnipulation) which solves the same problem as MOSAIC but in a non-modular way. By comparing the properties of MOSAIC and CARMA, we study the object–controller coupling in both a theoretical and experimental manner. More precisely, we show how object specialization in MOSAIC is actually quite sensitive to the learning parameters, and how CARMA avoids this issue.

The rest of the paper is organized as follows. We first describe the experimentation framework, as well as the MOSAIC and CARMA models. Our experiments begin with a validation of the capacity of both models to solve the MOM task. We then study the way it is solved in more detail, particularly regarding the controller specialization in MOSAIC. We finally show how the notion of object is encoded as part of the network activation structure in CARMA.

Experimental framework

We replicate the task defined by Gomi and Kawato (Gomi & Kawato, 1993) and applied in their subsequent papers (Wolpert & Kawato, 1998; Haruno, Wolpert, & Kawato, 2001), as closely as possible. It is the simulation of an arm that moves an object on a one-dimensional axis. The task is to move the object according to a given trajectory: the simulation has to choose, at each time step, a force to apply to the object.

The task becomes a MOM task when the object to be moved is changed at a fixed frequency during the simulation, and this change occurs in a single time-step. The amount of information available to the system is quite limited: the physical characteristics of the various objects, and the change frequency are unknown. This turns a simple linear equation system into a difficult cognitive task. Fig. 1 shows an example of the task.

In the simulation, any physical object is treated as a damped spring-mass system. Each object is thus defined by 3 parameters (M,B,K), with M the mass, B the viscous damping coefficient and K the spring constant.

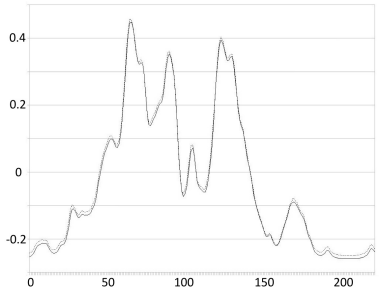

Figure 1: A sample desired trajectory to be followed (light) and the actual trajectory (dark) plotted against time.

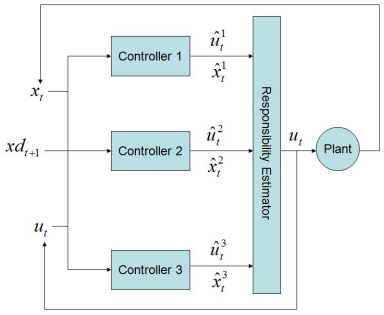

Figure 2: Global structure of MOSAIC with three controllers.

Time is treated as a discrete variable. We will use the following notations: for a given time index t, x_t is the object position, $\hat{x}_t$ the estimated object position, $\dot{x}_t$ is the object speed, O_t is the object identity, xd_t is the desired position, u_t is the applied force, and $\hat{u}_t$ is the estimated applied force.

A very simple plant model is used to compute, at each time step, the actual movement of the presented object O_t under the applied force u_t:

$$x_{t+1} = \frac{dt}{M}\left(u_t + \left(\frac{M}{dt} - B\right)\dot{x}_t - Kx_t\right) . \quad (1)$$

It is the only part of the simulation that actually knows the characteristics (M,B,K) of the objects O_t.

The MOSAIC model

The main idea of the MOSAIC model is to use multiple parallel controllers, each one suited for each particular object. Each controller is divided into three modules: the first encodes a Direct model, the second encodes an Inverse model and the last is a Responsibility Predictor, and is used to take into account visual information. Each of these modules is implemented using artificial neural networks (ANN).

Multiple controllers

At the highest level, the MOSAIC architecture is illustrated Fig. 2. Each controller is designed to predict the behavior of a particular object, but the actual control is done by all of them. At each time step, the responsibility of each controller is estimated. These responsibilities reflect the controllers' abilities to predict adequately the behavior of the current object. They are used in 2 ways: first, they weigh the contribution of each controller towards the final command (the controllers that predict well have greater control over the object), and second, they weigh the learning rate of each controller (the best predictors learn more and learn faster).

The responsibility λ_t^i of the i-th controller is computed by comparing the current position x_t with the position $\hat{x}_t^i$ estimated by controller i, as follows (Wolpert & Kawato, 1998):

$$\lambda_t^i = \frac{e^{-(x_t-\hat{x}_t^i)^2/\sigma^2}}{\sum_{j=1}^n e^{-(x_t-\hat{x}_t^j)^2/\sigma^2}} , \quad (2)$$

with σ a scaling parameter of this soft-max function.

Because the sum of all responsibilities is 1, they can be interpreted as probabilities: λ_t^i is the probability that the i-th controller is the best one to control the current object, according to the prediction errors. The σ parameter then regulates the competition between controllers.

Since the responsibilities gate both learning and forces, they are the heart of MOSAIC. The controller which was the best at predicting the object trajectory will have the highest responsibility, will learn more about controlling this object, which will help it predict more accurately, etc. Theoretically, it is supposed to make every controller specialize and converge to being the controller of a specific object.

Controller architecture

Each controller includes a Direct and Inverse model for a given object. The Direct model F predicts the future position given the current position, current speed and last applied control, while the Inverse model G computes the command to apply to go from a current position and speed to a desired future position:

$$\hat{x}_{t+1} = F(x_t,\dot{x}_t,u_t) , \quad (3)$$
$$\hat{u}_t = G(x_t,\dot{x}_t,xd_{t+1}) . \quad (4)$$

Each is implemented using linear ANNs (without hidden layers), with 4 nodes each. Indeed, for a single object, they approximate very simple equations with two unknown quantities and three parameters (M,B,K). The task of the Backpropagation algorithm is to adapt the weights of the network to give an implicit approximation of these parameters.

Visual modality

So far, the responsibilities are only based on the controllers' prediction error: it is feedback of a purely motor nature.

To more closely approximate the cognitive task of object manipulation, a third module is added to each controller, the Responsibility Predictor (RP), which simulates feedforward visual information. A visual representation is added to each object, in the form of a 3×3 matrix M_v of boolean pixels. The role of the RP is, given this visual representation, to predict the responsibility $\hat{\lambda}_t^i$ of its controller before any motion is performed. This feedforward responsibility estimation is merged

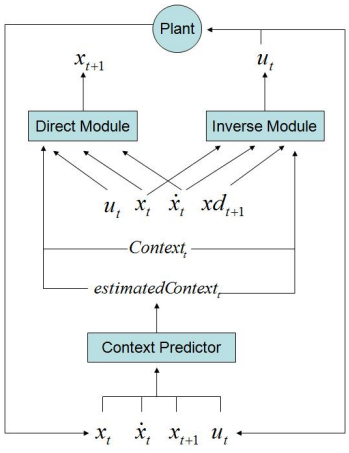

Figure 3: Global structure of the CARMA model. The direct, inverse and context predictor modules are four-layer MLPs.

with the motor feedback, and the responsibilities of Equation (2) are replaced by:

$$\lambda_t^i = \frac{\hat{\lambda}_t^i \times e^{-(x_t - \hat{x}_t^i)^2/\sigma^2}}{\sum_{j=1}^n \hat{\lambda}_t^j \times e^{-(x_t - \hat{x}_t^j)^2/\sigma^2}} \,. \tag{5}$$

The CARMA model

The global CARMA model takes the same inputs as MOSAIC $(x_t, \dot{x}_t, x_{t+1}, u_t, M_v)$, and is essentially structured like a single controller of MOSAIC (see Fig. 3): it is made of three modules, which are a Direct model, an Inverse model and a Contextual Predictor (CP). Each of these thus encodes knowledge relevant to several objects: therefore, they are more computationally complex than in MOSAIC. Whereas in MOSAIC, each module of a controller could be a linear ANN, in CARMA, each module is a four-layer Multi-Layer Perceptron (MLP), with an input layer, an output layer, and two hidden layers (with 10 and 2 nodes for the Direct and Inverse models, 10 and 5 nodes for the RP; the full CARMA model we used thus had 96 nodes).

Direct and Inverse models

The Direct and Inverse models have the same outputs as in MOSAIC, and the same inputs, augmented with two 3×3 matrices, which represent the actual visual input (real context) and the estimated visual input (estimated context). The real context input is the visual representation M_v of the manipulated object. Given the same input position, speed and force, this enables the Direct and Inverse models to output different values for different objects, according to this contextual input.

Context Predictor

The purpose of the CP is to identify the manipulated object, based on its dynamics. It uses motor feedback information to predict what should be the visual representation of the manipulated object. This estimated context can then be compared with the actual visual context; this comparison and the resulting difference drives the learning phase of CARMA. After convergence, this difference becomes very close to zero: the estimated and real context are almost always equal to one another.

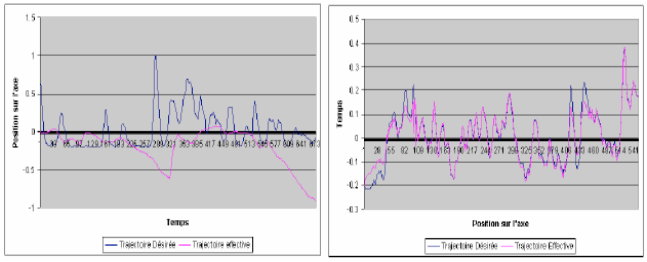

Figure 4: Solving the manipulation task with CARMA, before learning (left) and after learning (right). Three objects are switched every 20 time steps.

In some experiments, we also used the difference between the real and estimated contexts as a mechanism to handle illusions, where the system was fed a visual input which corresponded to a different object than the one actually manipulated. However, the details of these experiments are beyond the scope of this paper.

Experiments

In this section we first show that both systems can handle and solve the MOM task. We then analyze in more detail the way MOSAIC solves it. In particular, we show that MOSAIC's controllers do not become specialized for specific objects, except in special cases. We then study the mechanisms involved in CARMA for solving the MOM task.

Solving the task: experimental validation

MOSAIC and CARMA can both solve the MOM task without any problem. In Fig. 4 the results were recorded from CARMA controlling a set of 3 different objects, before and after the learning phase. Similar plots, obtained with MOSAIC, are not shown.

In order to prove that learning how to manipulate one object is not sufficient to manipulate all of them, we trained both systems on one given object, and, after convergence, gave them a different object (test object). We observed very low performance overall, as expected.

However, two cases could clearly be identified. If the system was trained on a lighter object than the test object, it would subsequently generate insufficient forces during test, which would not displace sufficiently the test object: the general trends of the trajectory would be followed, with large errors, large delays and slow convergence to the trajectory (Fig. 5, top). On the other hand, if the training object was heavier than the test object, the system would subsequently generate excessive forces, which would lead to overshoots and oscillating behaviors(Fig. 5, bottom). This was observed both in MOSAIC and CARMA.

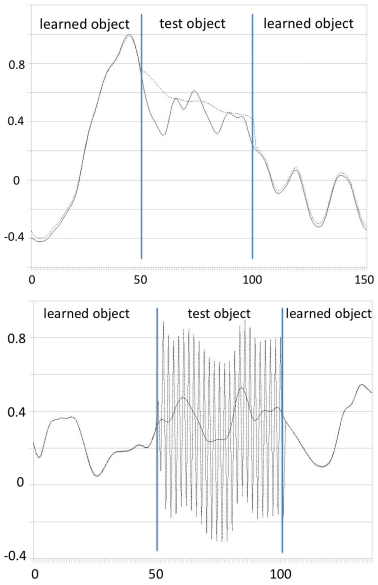

Figure 5: Using the learned controllers for an unknown, test object (from time step 50 to 100) either leads to damped and delayed control (top) when the test object is lighter, or oscillations (bottom) when the test object is heavier.

MOM with multiple controllers: MOSAIC

We then studied the behavior of MOSAIC for a MOM task.

The MOSAIC model relies on the property that, after convergence, each controller is specialized, in the sense that it should be responsible for the control of one and only one object. This property is well described (Haruno et al., 2001), but, unfortunately, we were not able to replicate it reliably. Indeed, according to our simulations, this specialization is not systematic: most of the time, it does not occur.

In typical cases, we observed that one controller acquires a large responsibility over all the objects, even if they have widely different physical characteristics (M,B,K). For instance, we presented the system with four objects (A to D), with different dynamics, and trained a MOSAIC system with four controllers (0 to 3). Despite the variability in the objects, we usually observed that one of the controllers was mainly responsible for most of the output commands, with marginal specialization in the remaining controllers. One typical case is shown Fig. 6.

Conditions for object specialization in MOSAIC

We discovered that object specialization in MOSAIC was quite sensitive to the values of the learning algorithm's parameters. We now detail them and explain their influence.

Learning rate The learning algorithm for the Inverse and Direct modules of each controller is the Backpropagation algorithm. If a controller is given a high responsibility for a short time, it learns a lot more than the other controllers and then already has an accurate control on all objects; we therefore used a low value (0.001).

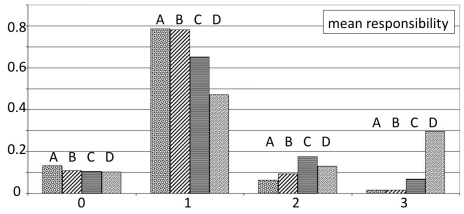

Figure 6: Mean responsibilities along a typical trajectory for a 4-controller MOSAIC (0 to 3) with 4 objects (A to D): here, controller 1 (second block of bars) takes care of most of the control for all objects, while controllers 2 and 3 are marginally specialized for objects C and D, respectively. Object A: $(M = 1, B = 2, K = 8)$, object B: $(M = 1, B = 8, K = 1)$, object C: $(M = 3, B = 1, K = .7)$, object D: $(M = 8, B = 2, K = 1)$.

Object switching frequency This frequency has a crucial importance during the learning phase. If the frequency is too low, the situation is similar to sequential training: large training on one object, then on another one. In this case one controller becomes perfect for one object, and is also better for the other objects than untrained controllers: this is the property we illustrated previously (see Fig. 5). In our simulations we switched objects frequently, every 20 time steps.

Controller competition parameter σ This parameter seems to be key for controller specialization. Unfortunately, the way it is defined in MOSAIC is unclear: it is only said to be "tuned by hand over the course of the simulation " (Haruno et al., 2001, 2211). We therefore investigated three cases.

If σ is set to a low value, the competition is strong between controllers: as soon as one controller specializes for one object, as it is also better than untrained controllers on the other objects (see Fig. 5), it wins control over all objects. Moreover, only one controller is active at a time: the system becomes similar to a mixture of experts system (Jacobs, Jordan, Nowlan, & Hinton, 1991).

If σ is set to a high value, the cooperation is strong between controllers: the responsibilities are so well distributed that almost no specialization appears. All controllers have nearly the same responsibilities so they share the control of the objects. Despite this, the manipulation error remains small.

The last case is to have σ vary during training, and more precisely, decrease over the training period. Indeed, with an initial cooperation and shared control between controllers, they all quickly learn the main characteristics of the motion equation, and the main aspect of control: apply a positive force when the object needs to go up, a negative force otherwise. When this is trained into all controllers, then σ can be slowly decreased so that controllers, in turn, pick more precise characteristics of the physical behaviors of the objects. Finally, σ should decrease over time, but not in a linear way since the convergence of the ANNs is not linear. When it is correctly tuned, a specialization can be observed (Haruno et al., 2001). Unfortunately, the function $\sigma(t)$, according to our

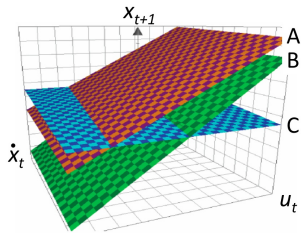

Figure 7: Dynamics for 3 objects (A to C) with different characteristics (M,B,K). The axes are: speed $\dot{x}_t$, applied force u_t and next position x_{t+1}. A fixed starting position is assumed. Each plane corresponds to one object.

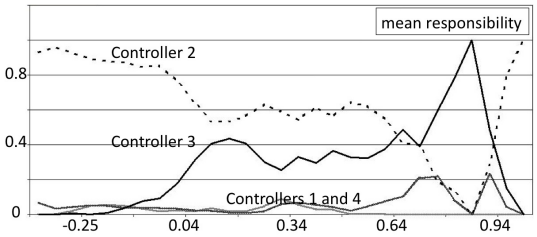

Figure 8: Responsibilities for 4 controllers in MOSAIC, plotted against the desired position xd_t.

simulations, also appears to be dependent of problem specific factors, including the number of presented objects and their characteristics (M,B,K); we do not foresee an easy way in which $\sigma(t)$ could be automatically defined in order to be suitable for a given instance of the MOM task.

What is learned by controllers in MOSAIC?

A close inspection of the physical manipulation problem shows that some structural properties are the same for all objects: for instance, discrimination between the pulling cases (negative force) and the pushing cases (positive force).

Further investigations also show that some objects with different physical characteristics become indistinguishable for some trajectories. For instance, consider two objects with the same mass M and spring constant K but with different damping factors B_1 and B_2: when the speed along the trajectory is small, these two objects behave similarly, and a single controller can easily control both. On the other hand, when the speed is high, the forces to output are different, and two controllers are needed. This is illustrated Fig. 7: we plotted the motion equation (1) for three different objects. In order to represent it on a 3D plot, we set a fixed starting position x_t. In this projection of the Space Of Dynamics (SOD), we can observe that objects are intersecting planes. At the intersections, the objects are indistinguishable.

Because objects are indistinguishable for some trajectories, we hypothesized that controllers in MOSAIC would not become specialized for specific objects, but rather, for object–trajectories combinations. We thus plotted the controller responsibilities after learning against the trajectory characteristics. We show Fig. 8 the responsibilities for 4 controllers (0 to 3) manipulating 3 objects (A to C) as a function of the desired position xd_t: we observe that when $xd_t < 0$, controller 2 takes almost full control, that there is a shared control between controllers 2 and 3 for $xd_t \in [.1,.7]$, and that controller 3 is specialized for $xd_t \in [.7,.9]$, *independently of the object being manipulated*. Therefore, it appears that MOSAIC controllers indeed specialize for motion subspaces.

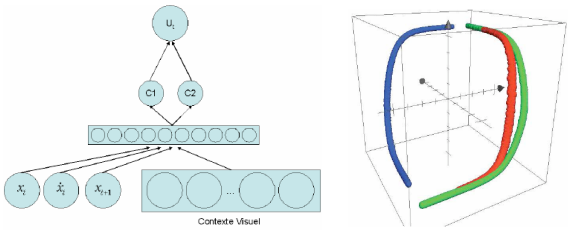

Figure 9: On the left, the CARMA Inverse module with an additional two-node layer for investigating the structure of the learned network. On the right, activity plot of the Inverse module; the axes are the values in the 2D added layer (X and Y) and the output value of the network (Z).

MOM with a single controller: CARMA

Since CARMA solves the MOM task with a single controller, and because it does not encode objects in its structure, we studied the way different objects were represented in the Direct model and Inverse model ANNs after learning.

We first quickly describe a new method of plotting the activation of a MLP neural network, and then use it on CARMA to investigate its internal representation of objects.

In a MLP, each layer is a transformation of the input space that can have a different dimensionality. If we add a two-node layer to the network, it is possible to extract a two-dimensional transformation of the input space and plot it. To generate the plotting data, we first train the network, then disable learning, submit to the network a random input activity on the nodes of interest, propagate it through the network, and, finally, record the activity of the two-dimensional hidden layer. This process is iterated until enough data is collected to have a good representation of the input space.

We used this method on CARMA's Inverse module (Fig. 9, left). By logging the activity of the two-dimensional hidden layer activity and the one-dimensional output layer we can draw a 3D plot of the function approximated by the whole module. In the case of a network trained on multiple objects, this representation gives information about the internal representations of objects. The most interesting result is that the function is fragmented: multiple long shapes are partially merged (Fig. 9, right). The number of shapes is equal to the number of objects learned by the system. With this method we get a clear representation of what an object is for CARMA: the concept of object is no longer a structural property of the model, it a contiguous set of activities in the set of possible activations in the network.

In the activity plot, the proximity of the shapes provides useful information. Some of them are merged, meaning that the objects are indistinguishable based on their dynamics. Furthermore, only one factor modifies the shape positions in the plot: the visual representation of the object. In other words, CARMA produces different outputs for different visual representation. This means that CARMA learns the motion equation, and uses the visual representation of the object as its parameters. Thus, the characteristics (M,B,K) are encoded in the internal visual representation. This encoding is the key point of the Context Predictor module.

Recall that the Context Predictor inputs are from the space of object dynamics, and its output is an estimated visual context: the CP learns a mapping from object dynamics to the visual representation space. In other words, it associates physical behaviors of objects with their appearances. The module never computes the (M,B,K) parameters explicitly, but encodes these in a visual space.

There is an interesting analogy with the way humans are not able to exactly ascertain the mass of an object. It is easy to know that one object is heavier than another, but very difficult to provide a precise estimation of a mass. Indeed, humans probably encode mass in a non-numeric space which would be a mixture of volume, aspect, dynamic experienced by motor experience, etc.

To further study this analogy, it would be fruitful to train the system and verify whether and how similar visual representations are associated with objects with similar dynamics; in other words, study the metrics of the transformation between the visual space and the space of the (M,B,K) parameters. With manually designed visual representations (*e.g.* objects with very similar visual representations but very different dynamics), it would be possible to test the predictions made by the Contextual Predictor.

Discussion

We presented and tested two systems, MOSAIC and CARMA, designed to solve the Multiple Object Manipulation task. The main difference between them is the modularity hypothesis: MOSAIC assumes that objects are encoded in a spatial way, into the model structure; whereas CARMA builds a function which handles all objects. Our experimental study has shown that, in MOSAIC, the controller–object association is not systematic and mainly relies on a human tuned parameter. Most of the time, controllers specialize on complex mixtures of trajectory, motion and object, which we have shown to be a central property of the CARMA model.

The non-modular approach was criticized by the authors of MOSAIC. According to them, for instance, a single controller would be too computationally complex. Actually, for the same problem, CARMA uses less neurons than MOSAIC. Indeed, in CARMA, the computational power comes from the number of nodes in hidden layers, while in MOSAIC, complete Direct and Inverse models are duplicated for each additional object. For instance, our CARMA implementation, with 10 hidden nodes in the Direct and Inverse models, and a total of 96 nodes, solves the MOM task with 10 objects. In MOSAIC, it would require 8*10 objects + 20 (for the RP) = 100 neurons. We believe that the difference would grow for additional objects, as CARMA with a few more nodes would treat a large number of additional objects (as we illustrated experimentally but did not expose in detail here).

They also suspected a slow adaptation to context variation; however, there is no delay in CARMA since the context is what defines the output of the system. The last point is the sensibility to catastrophic unlearning, which we did not study in this paper, but which has been solved elsewhere on similar single controllers, by a method that can easily be adapted to CARMA (Ans & Rousset, 2000).

Studying MOSAIC has implications beyond the scope of pure mathematical modeling. Indeed, the modularity hypothesized in MOSAIC – one controller for one object, and therefore, a spatial, structural encoding of objects in the global controller – is taken as a starting point of some recent brain imagery studies (Imamizu et al., 2000; Imamizu, Kuroda, Miyauchi, Yoshioka, & Kawato, 2003; Ito, 2000). Therefore, equivalents of this structural object endoding are looked for in the biological substrate; there is here the risk of an interpretation bias, resulting from taking for granted a model which is too specific.

References

Ans, B., & Rousset, S. (2000). Neural networks with a self-refreshing memory: Knowledge transfer in sequential learning tasks without catastrophic forgetting. *Connection science*, *12*, 1–19.

Gomi, H., & Kawato, M. (1993). Recognition of manipulated objects by motor learning with modular architecture networks. *Neural Networks*, *6*, 485–497.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Computation*, *13*(10), 2201–2220.

Imamizu, H., Kuroda, T., Miyauchi, S., Yoshioka, T., & Kawato, M. (2003). Modular organization of internal models of tools in the human cerebellum. *Proceedings of the National Academy of Science (PNAS)*, *100*(9), 5461–5466.

Imamizu, H., Miyauchi, S., Tamada, T., Sasaki, Y., Takino, R., Pütz, B., et al. (2000). Human cerebellar activity reflecting an acquired internal model of a new tool. *Nature*, *403*, 192–195.

Ito, M. (2000). Neurobiology: Internal model visualized. *Nature*, *403*, 153–154.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.

Kawato, M. (2008). From 'understanding the brain by creating the brain' towards manipulative neuroscience. *Phil. Trans. R. Soc. B*, *363*, 2201–2214.

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*(7-8), 1317–1329.