**Bi-directional Modularity to Learn Visual Servoing Tasks**

Gilles Hermann, Patrice Wira, Member, IEEE and Jean-Philippe Urban

Abstract—This paper shows the advantage of using neural network modularity over conventional learning schemes to approximate complex functions. Indeed, it is difficult for artificial neural networks like Kohonen extended maps to converge toward an efficient and adequate solution when the dimensionality of the input and output spaces are high. Associated to an appropriate learning technique, modularity in neural networks is able to overcome the high dimensionality of the input/output space by decomposing it into different intermediate spaces of reduced dimensionality. The decomposition results in independent neural modules. The efficiency of this learning technique will be enlightened with a visual servoing application. In this application, the relationship between the visual features issued from a stereoscopic vision system and the angles of a 5 DOF-robot will be learned and approximated. Simulations have been conducted and clearly show that this complex, non-linear, and high dimensional function can be learned efficiently with the neural network modularity approach. Moreover, we show through these simulations that neural modules can be re-utilized, thus reducing the convergence time of the learning and the memory requirements.

I. INTRODUCTION

Visual servoing tasks, like estimating the angular pose of a 5-DOF (Degrees Of Freedom) robot with only visual information, are modern and challenging problems. Several approaches can be used to determine the model of such a system [4]. In this work, we propose to use a neuromimetic approach, based on extended kohonen maps [11], [1], [5].

These ANNs (Artificial Neural Networks), efficient in numerous applications, are not well suited for approximating non-linear and high-dimensional functions like the ones in robot vision systems and more especially in visual servoing for the control of a robot arm. We consider an application composed of a stereoscopic vision sensor with 4 DOF to orientate two cameras and a 5-DOF-robot arm. The wrist and the end-effector point of the robot arm are extracted from the image data and compose with the cameras angles the 12 elements of the input vector of the learning system. The learning of this function with 12 inputs can not be guaranteed in term of convergence and precision with a single ANN, i.e., a Kohonen map. We propose to replace it by several neural modules resulting from a decomposition where each module consists of a Kohonen map of smaller sizes facilitating the learning. The whole modular approach thus keeps the properties and advantages of the self-organizing maps [10]. New intermediate variables, necessary for the learning of the modules, are introduced. The local learning of each module also implies an inverse data flow which will be described in the next section.

The modules resulting from the decomposition of the global function are organized according to two constraints. First, each module has to learn and approximate a real variable having a physical interpretation. Secondly, in order to limit the memory requirements and to guarantee the convergence of its learning, the input vector of each module is limited to less than three.

This paper is structured as follow. In the next section, we describe the ANNs that will be used, with their advantages and drawbacks. Section III presents the different existing learning architectures by insisting on the one we retain. In Section IV, we develop a decomposition that learns the non-linear function of a robot-vision system. This decomposition leads to the estimation of the angular pose of a robot with only visual information. Results are presented in Section V and conclusions are drawn in Section VI.

II. EXTENDED KOHONEN MAPS (SOM-LLM)

The neural network adopted for the implementation of a neural module is the SOM-LLM (Self-Organizing Map-Local Linear Map)(see Figure 1)[2]. This neural network is a combination of the extended self-organized Kohonen map [8], [11] and ADALINE networks. The SOM-LLM has been chosen for its simplicity, for its implementation facilities, and for its topological properties (neighborhood and competition between neurons). The main objective of the SOM-LLM is to approximate any transformations by linear local estimations.

The SOM-LLM is composed of a grid of neurons for the discretization of the input space, a grid of neurons for the discretization of the output space, and ADALINES associated to each neuron from the output map to compute a linear local response.

For each input vector (stimulus), the neurons of the input map compete to be the best representation of the input $x_k$. The weights of the winner, $w_s^{(in)}$, are those minimizing the distance with the input:

$$s = \arg\min_{s} \|x_k - w_s^{(in)}\|.$$  

The weights $\Delta w_s^{(in)}$ and $\Delta w_s^{(out)}$, respectively in the input and in the output maps, are adapted using the two following equations:

$$\Delta w_s^{(in)} = w_s^{(in)} + \varepsilon h_s \left( x_k - w_s^{(in)} \right),$$

$$\Delta w_s^{(out)} = w_s^{(out)} + \varepsilon' h_s' \left( u - w_s^{(out)} \right),$$
where $\varepsilon$ and $\varepsilon'$ are the learning rates, $h_s$ and $h_s'$ the neighborhood functions, and $\hat{u}$ the desired output.

Thus, the output of the network is expressed by:

$$y = w_{s}^{(out)}u + A_s(x_k - w_{s}^{(in)})$$

(4)

where $A$ is the ADALINE weight vector.

III. MODULARITY IN NEURAL NETWORKS

Modularity in neural networks corresponds to a functional neural network organization with several blocks composed of independent ANNs. Various possible organizations can be considered, including the modular architectures presented by Figure 2. Among these configurations there are: the hierarchical structure (like our SOM–LLM for example), the parallel structure and the serial structure.

A. The parallel architecture

In a parallel architecture, all the modules process the data simultaneously. Each module is independent. The output can be given by all the modules or not. Such a combination of networks can either make it possible to break up a task into sub-tasks, and one will speak about true modularity, or use several networks for the same estimate and thus to make improvements in term of performance and reliability, as the mixtures of experts do it [7]. It is this first approach which tends to locally reduce the dimensionality of the problem on which we focus here.

B. The serial architecture

In a serial architecture, a task is broken up into successive sub-tasks. The inputs are processed as they cross the various modules of the architecture. Indeed, the outputs of the previous modules are used as inputs of following modules. Connection values are inevitably introduced by the serial configuration. These connection values are necessary to train each module. If these values are inaccessible, setting up structures of training is needed.

C. Discussion

Each modular architecture has advantages and drawbacks. Indeed, parallel learning architectures are easy to train (inputs and outputs are available) and allow simultaneous processing of the tasks. Nevertheless, parallel learning architectures do not reduce the size of the input vector id the input data are correlated for a specific task. Although we use the two types of modular learning architectures, we focus on serial modular architectures in the next section. In fact, serial architectures permit to reduce the size of the input vector of each module. On the other hand, these architectures require to define some new learning paradigms.

IV. MODULAR LEARNING

Serial architectures require a particular technique of learning. For other modular architectures, the learning remains traditional [3].

The serial decomposition of a task introduces intermediate variables (or inner variables), the connection values mentioned above. Those are necessary for the learning of the various modules that compose the modular architecture. These variables often are neither known, nor measurable. They thus must be estimated. We introduce, with this intention, an inverse flow and we obtain a bi-directional learning architecture [9].

We define the bi-directional learning with the example, presented by Figure 3. The relation to be learned, $F()$, links the space of the inputs $x$ to the space of the outputs $y$.

$$y = F(x_1, x_2)$$

(5)

where, for example, $y = y$, $x_1 = [x_1, x_2]^T$ and $x_2 = [x_3]$. The resulting decomposition is made up of two modules $M1$ and $M2$, respectively estimating the functions $f_1()$ and $f_2()$. It is the direct flow. The module $M1$ takes part of the vector of entries $x$ to make a $z_1$ estimate. This estimate as well as the remainder of the vector of entries are used by the module $M2$ to make the $\hat{y}$ estimate. The function $F()$ can then be rewritten in the following manner:

$$F(x_1, x_2, x_3) = f_2(x_3, z)$$

(6)

with : $z = f_1(x_1, x_2)$.

The learning of the function $F()$ and thus of its sub-functions is supervised. Each neural module needs a number of selection inputs/desired outputs pairs. Consequently, we need the variable $z$ to be able to learn the first module. These desired outputs will be provided by a third module $M2^{inv}$, inserted in the modular structure in the opposite flow. This module makes a second estimate of the variable $z$, it generally takes its inputs in the outputs space. It is necessary that the function $z_2 = \varphi(y)$ exists. The association of these direct and inverse flows constitute the bi-directional architecture. In same manner that the output of the module $M2^{inv}$ is used as a model for the module $M1$, the output of the module $M1$ is used as a model for the module $M2^{inv}$. The learning of these two modules is done simultaneously.
During the learning, the two estimates of the variable \( z \) must converge towards the same representation.

The inner representation of \( z \) is not unique. It corresponds to the joint solution of the two functions \( f_1() \) and \( g() \). One must keep in mind that the task is decomposed into several modules, where each represents a physical value. \( z \) is not exactly a physical value but a function of a physical value, because of the modular learning.

The adaptation rules which we adopt to learn each module introduce additional constraints which prevent the estimated variables \( \hat{z}_1 \) and \( \hat{z}_2 \) to converge toward a constant trivial solution, non representative of the variable to be estimated. Under these constraints the inner representation of \( z \) converge towards null average and unity variance. These constraints are expressed in the following equations by the terms \( p_t \), which represents an estimate of the average of the response of the first network, and \( V_t \), which is a coefficient of correction of the variance.

The adaptation models of the weights are defined by the following equations, according to the \( \epsilon_z \) error between the two estimates of \( z \):

\[
\epsilon_z = \hat{z}_2 - \hat{z}_1 ,
\]

\[
w_{s1}^* = V_t \left( z_1 + \epsilon_z - p_t \right) ,
\]

\[
w_{s2}^* = \hat{z}_2 - \epsilon_t ,
\]

\[
p_{t+1} = p_t + \gamma_p \left( \hat{z}_1 - p_t \right) ,
\]

\[
V_{t+1} = V_t + \gamma_v V_t \left( 1 - (V_t \hat{z}_1)^2 \right) ,
\]

\( \gamma_p \) and \( \gamma_v \) are time-varying adjustment factors. These coefficients must be rather large at the beginning of the learning, in order to take into account averages and variances of large variations of the weights of the outputs map. On the contrary, at the end of the learning, these coefficients must be small, possibly null, in order to avoid to generate noise, since the output variance is almost stable.

An additional advantage in these learning constraints is that they normalize the inputs of the other modules, i.e., \( M2 \), so that all the inputs are taken into account in the same way during the selection phase of the winner neuron.
V. PROPOSED DECOMPOSITION FOR THE VISUAL SERVOING TASK

In this section, a visual servoing task will be solved in order to illustrate the bidirectional modular learning process (Figure 4).

A. Robot-vision setup

We wish to learn the relation between information issued from a vision system and from a robot-like arm. The vision system is composed of two individually directional cameras (the pictures size are about 640 × 480 pixels, the pixel size is 11 × 13 µm). Information available are the coordinates of the wrist and the robot-like arm effector, projected in each of the two images, as well as the two angles (pan and tilt), which make it possible to direct the two cameras. The robot used is a four axes robot with five DOF. The first angle, \( \theta_1 \), enables the arm to turn around its base. The coplanar angles \( \theta_2 \), \( \theta_3 \) permit to place the wrist. The angles \( \theta_4 \) and \( \theta_5 \) direct the effector. A schematic view of the robot is presented on Figure 5. The four axis length are respectively: \( l_1 = 680 \text{mm} \), \( l_2 = 400 \text{mm} \), \( l_3 = 480 \text{mm} \) and \( l_4 = 160 \text{mm} \).

B. Neural decomposition

The selected decomposition is based on the geometry of the robot-vision system (see Figure 6). Since one of the objectives of the decomposition is to reduce the number of the entries of the different modules, the first step of our decomposition is to reduce the number of visual information. Usually, a point in the 3D space is represented by two coordinates and two angles for each image, leading to a total of eight visual information. The solution chosen to reduce this number of inputs is to learn an alternative configuration from the robotic head which consists in centering the observed object. In this configuration, the centered object is perfectly defined by the angles which direct the two cameras, and therefore with only four information. We will speak about \( \alpha \)-centered, \( \alpha_c \), to name the new angles which direct the centered cameras. This centering is done by a first module composed of four neural networks, each making the estimate
of one angle of the "centered cameras" configuration. Each one of these networks of neurons has three inputs. As already mentioned, we will take care that each module of the decomposition does not have more than three inputs. The inputs of the four neural networks are:

- The left camera vergency angle \( \alpha_{vg} \), the left camera tilt angle \( \alpha_{tg} \), and \( X_g \), the left object image projection to be centered for the \( \alpha_{cg} \) estimate.
- The left camera vergency angle \( \alpha_{vg} \), the left camera tilt angle \( \alpha_{tg} \) and \( Y_g \), the left object image projection to be centered for the \( \alpha_{ctg} \) estimate.
- The right camera vergency angle \( \alpha_{vd} \), the right camera tilt angle \( \alpha_{td} \) and \( X_d \), the right object image projection to be centered for the \( \alpha_{ctd} \) estimate.
- The right camera vergency angle \( \alpha_{vd} \), the right camera tilt angle \( \alpha_{td} \) and \( Y_d \), the right object image projection to be centered for the \( \alpha_{ctd} \) estimate.

The second step of the decomposition is to learn the relation between new image information outputs from the first module and the observed point three-dimensional coordinates, \( X_i, Y_i \) and \( Z_i \). The second module is composed of only one neural network. It takes three inputs: \( \alpha_{cvg}, \alpha_{ctg}, \) and \( \alpha_{vg} - \alpha_{ctd} \). The learning examples can result simply from the direct model of the robot-like arm.

The \( \alpha \)-centered are inner variables introduced by the decomposition. In order to permit the learning of the first module, and to provide examples of trainings, we introduce a third module into inverse flow. This module is composed of a three inputs neural network : \( X_i, Y_i \) and \( Z_i \) and made estimates of \( \alpha_{cvg}, \alpha_{ctg}, \) and \( \alpha_{vg} - \alpha_{ctd} \).

This bi-directional modular architecture is used to define alternately the coordinates of the wrist and the effector. To obtain an effective learning, it is necessary that the learning examples cover all the possible positions of the effector and the wrist.

The estimation angle of the robot-like arm’s first three axes, \( \theta_1, \theta_2 \) and \( \theta_3 \) are carried out by a neural network (module 3), whose three inputs are the estimates of the coordinates of the wrist, \( X_p, Y_p \) and \( Z_p \). It is interesting to note that wrist \( \alpha \)-centered information also make possible to determine the first three angles of the robot-like arm easy to design.

The wrist and effector 3D coordinates are used to calculate the length wrist-effector projected on the horizontal plane and consequently used to estimate \( \theta_4 \). The association of a calculated length with the preliminary estimate of \( \theta_4 \) permits to estimate \( \theta_5 \) (module 5).

VI. SIMULATION RESULTS

This section shows the learning results of the different networks of the modular architecture. The neighborhood function and the learning rates evolve according to the rule suggested by Ritter in [11].

A. The alternative configuration learning

This first module allows to learn a new cameras configuration, reducing the number of visual informations necessary to characterize a point in the 3D space. The desired outputs of this module are provided by a neural network placed in the inverse flow. The direct and the inverse modules are two 3D networks of 1100 neurons each one. Figure 7 shows the evolution of the response error during the learning of the variable \( \alpha_{cvg} \) according to the number of presented examples. In this figure, the continuous line represents the average windowed error on 100 examples, the "o" represents the maximum error among 100 examples. The characteristic curve that we observe confirms the learning. The very large error at the beginning decreases quickly after 5000 iterations. At the end of the learning, i.e., after 10000 examples, the average of the estimation error tends towards 0.01 degrees and is close to a maximum of 0.24.

Figure 8 shows the desired output in function of the estimated output: \( \alpha_{cvg} = f(\hat{\alpha}_{cvg}) \). We can see on this figure that the estimated variable is representative of the desired variable. The relation is bijective. However we can note
that the ranges of $\alpha_{cg}$ and its estimate $\hat{\alpha}_{cg}$ are not the same. This comes from the constraints on the average and the variance applied during the bi-directional learning with (7-11).

The representation for the three other variables, i.e., $\alpha_{vd}$, $\alpha_{tg}$, and $\alpha_{td}$ are very similar to those for $\alpha_{vg}$. Table I summarizes the average and maximum errors of first module estimations.

<table>
<thead>
<tr>
<th>Estimated values</th>
<th>Mean error (in degrees)</th>
<th>Max error (in degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{\alpha}_{cg}$</td>
<td>0.01</td>
<td>0.24</td>
</tr>
<tr>
<td>$\hat{\alpha}_{vd}$</td>
<td>0.02</td>
<td>0.36</td>
</tr>
<tr>
<td>$\hat{\alpha}_{tg}$</td>
<td>0.01</td>
<td>0.24</td>
</tr>
<tr>
<td>$\hat{\alpha}_{td}$</td>
<td>0.01</td>
<td>0.67</td>
</tr>
</tbody>
</table>

B. Estimation of 3D coordinates

The estimations of the first module are used by the second one in order to determine the 3D space coordinates of the centered object. The desired outputs of this module could be furnished either by a module in the inverse flow, or by using the direct model of the arm-like robots. This model is easy to establish.

The used neural network has three inputs: $\hat{\alpha}_{vg}$, $\hat{\alpha}_{tg}$, and $\hat{\alpha}_{vd} - \hat{\alpha}_{td}$, and three outputs: $X_i$, $Y_i$ et $Z_i$. This network is composed of 1100 neurons. The estimated error of the $Z_i$ value during the learning is shown in the Figure 9. It corresponds to the height in the 3D space. Table II shows the learning results of all estimated values by this module.

<table>
<thead>
<tr>
<th>Estimated values</th>
<th>Mean error (in mm)</th>
<th>Max error (in mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_i$</td>
<td>0.32</td>
<td>38</td>
</tr>
<tr>
<td>$Y_i$</td>
<td>0.21</td>
<td>26</td>
</tr>
<tr>
<td>$Z_i$</td>
<td>0.27</td>
<td>17</td>
</tr>
</tbody>
</table>

In order to perform an efficient learning, the redundancies should be avoided. Figure 10 shows the error in estimating $\theta_2$, and Table III presents the mean square error and its max. in estimating the angles $\theta_1$, $\theta_2$ et $\theta_3$. This are acceptable because the ranges of $\theta_1$, $\theta_2$, and $\theta_3$ are about 90 degrees.

These angles allow to fully determine the angles of the wrist to position it in the 3D space and this is done with an error less than 1mm in mean and less than 1cm in max. after the training period. This precision naturally depends on the precision in estimating $X_i$, $Y_i$ et $Z_i$. 
TABLE III

<table>
<thead>
<tr>
<th>Estimated values</th>
<th>Mean error (in degrees)</th>
<th>Max error (in degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{\theta}_1 )</td>
<td>0.03</td>
<td>0.26</td>
</tr>
<tr>
<td>( \hat{\theta}_2 )</td>
<td>0.07</td>
<td>1.47</td>
</tr>
<tr>
<td>( \hat{\theta}_3 )</td>
<td>0.06</td>
<td>1.25</td>
</tr>
</tbody>
</table>

D. Estimation of \( \theta_4 \)

After estimating the angles \( \theta_1, \theta_2 \) and \( \theta_3 \), the angle \( \theta_4 \) is calculated. Modules 1 and 2 are used to estimate the coordinates of the wrist and of the end-effector and the estimation of \( X_i \) and \( Y_i \) are only necessary.

The estimated orientation \( \theta_4 \) can be expressed by the following expression:

\[
\theta_4 = \cos^{-1}\left( \frac{\sqrt{(X_P - X_E)^2 + (Y_P - Y_E)^2}}{l_4} \right),
\]

(12)

with \( l_4 \) the length of the effector of the robot.

Estimating \( \theta_4 \) with (12) is achieved with a mean square error of 0.17 degrees and with an error max. of 14 degrees. The precision in estimating \( \theta_4 \) mainly depends on the precision in estimating the variables implied in (12).

As an alternative, we also show that \( \theta_4 \) can be learned. Indeed, we propose to use a neural module to learn the relationship between \( \theta_4 \) and the distance \( \sqrt{(X_P - X_E)^2 + (Y_P - Y_E)^2} \). This learning allows to estimate \( \theta_4 \) with an error less than 7 degrees which is better than the estimation obtained with expression (12).

E. Estimation of \( \theta_5 \)

We propose to estimate the angle \( \theta_5 \) with a neural module with two inputs: an inner variable issued from the module that calculate \( \theta_4 \) and the estimation of \( \theta_1 \) issued from module 3. The precision in estimating \( \theta_5 \) also depends on either the estimation of \( \theta_1 \) and the previous inner variable. Figure 11 shows the error in estimating \( \theta_5 \) and one can see that the mean square error is less than 0.13 degrees and that the max. error is less than 4 degrees at the end of the training period. This is acceptable because the range of \( \theta_5 \) is about 90 degrees.

F. Discussions

The different simulations show that complex and non-linear functions can be learned with a modular bi-directional learning architecture based on modules composed of SOM-LLM neural networks. This modular bi-directional learning architecture enforces the learning of each module to converge. The learning of the whole function without a modular approach, i.e., with a single neural network, is not possible in part because of a high number of variables (inputs) [6]. Simulations have been done and show that the SOM-LLM with 12 inputs is not able to learn the robot-vision system function even with a high number of neurons.

The results achieved with the modular bi-directional learning architecture depend on several parameters, in particular the size of the neural networks, the quality of the input data, and the data sets used for the training. The most critical one is the size of the output vector, which influences the convergence of the learning. Moreover, the modular architecture is well suited for an on-line learning if the size of the neural networks are small (thousands of neurons).

The estimated angle with the modular bi-directional learning architecture allow to control the robot arm and to reach a position in the 3D space with an error less than 1mm. These results are considered to be very good compared to the fact that the 3D scene is observed with a resolution of the same order with the two cameras (one pixel in both images corresponds to 1mm in the 3D space). Moreover, the max. error can be reduced by constraining the working space of the robot arm.
VII. Conclusion

We have presented a robust modular neural architecture that is able to learn complex systems. Learning robotic tasks is not straightforward. Nonlinear, often with a great number of degrees of freedom, it requires a lot of computational power, lots of training data, and the convergence may then not be enforced. We chose modularity, to decompose the complexity of a problem. A high dimensional fully connected neural network is replaced by a set of well organized modules which can be more easily trained. The main contribution of the work is a formulation that makes a set of neural modules able to converge and to learn any complex system.

Negative aspects of the use of artificial neural networks remain, however, in that they present problems in terms of stability and approximation analysis, and furthermore, it is often difficult to choose the modular network architecture. Even though, the proposed modular neural network is very well suited for on-line applications.

The modular neural architecture and the bi-directional learning have been evaluated with a visual servoing task. The non-linear relationship between visual features and robot angles has been efficiently estimated with several neural modules of reduced sizes. Compared to a single neural network, the modular neural architecture reduces the convergence time and the memory requirements and allows to position a 5 DOF robot in the 3D space with high precision.

Future study will concern the possibility of using other constraints in the modular neural architecture to enforce the convergence toward representative internal representations. Finally, further work is needed to understand the structural properties of the modular network and the plant under control.

REFERENCES