Integrating representations for learning higher-order correlations in a brain-inspired cognitive framework

Martin Heracles, Alexander Gepperth, Jannik Fritsch, Christian Goerick

2008

Integrating representations for learning higher-order correlations in a brain-inspired cognitive framework

Martin Heracles, Alexander Gepperth, Jannik Fritsch, Christian Goerick

***Abstract*— An important biological mechanism for learning in the human brain is Hebbian synaptic plasticity. However, Hebbian learning is restricted to direct synaptic connections between two neural populations, hence cannot directly account for learning that involves multiple neural populations. In order to overcome this limitation, we propose to combine Hebbian learning with self-organizing maps. More specifically, multiple lower-level representations are mapped onto self-organizing maps in an unsupervised process, thus forming higher-order representations of combinations of the lower-level representations. This way, learning that involves multiple neural populations can be reduced to learning that involves only two (higher-order) neural maps, which makes Hebbian learning applicable again.**

In order to demonstrate the validity of our approach, we consider a simple case of learning that involves three lower-level representations, namely, learning the interrelationship between the 2D size of objects in monocular camera images, their object class as obtained by an object classifier, and their 3D distance to the camera. We present a technical system instance of the proposed system architecture that is able to learn this interrelationship at run-time and, after learning has finished, to predict the 3D distance of objects from their 2D size and their object class. Experiments using both simulated and real-world image streams confirm the quality of these predictions and support our hypothesis that the proposed combination of Hebbian learning with self-organizing maps can serve as a generic mechanism for system-wide unsupervised learning.

I. Introduction

Despite recent advances in research fields that strive for autonomous intelligent systems, examples of which include robotics and intelligent vehicle research, state-of-the-art technical systems are still far from being comparable to the performance of humans. This is in particular true when it comes to highly complex, dynamic real-world environments such as traffic scenes in an inner-city environment, for example. One of the most striking capabilities of humans is the high degree of robustness they exhibit at different levels, be it robustness to unreliable percepts, to difficult environmental conditions such as rain or direct sunlight, or the ability to cope with unforeseen situations. To our mind, any autonomous intelligent system will need such kind of robustness when operating in unconstrained environments like this.

We think of this high degree of robustness as the result of an incremental learning process, similar to the way young children develop. This process involves observing the environment, discovering interrelationships and learning from experience, which we consider to be largely unsupervised. While there are certainly many things that cannot be learned in an unsupervised manner but require explicit teaching, we advocate the use of unsupervised learning methods to the greatest extent possible because this significantly reduces the effort one has to take in creating complex large-scale systems.

Looking at the neural basis for learning in the human brain, one important mechanism is Hebbian synaptic plasticity [1]. Thanks to this plasticity, it is possible to learn correlations with respect to the activity of different neural populations. However, Hebbian learning is inherently limited to correlations between pairs of neural populations. It therefore cannot directly be used for learning correlations between multiple populations. In order to overcome this restriction, we propose to combine Hebbian learning with self-organizing neural maps [2].

Essentially, the approach we propose amounts to the formation of increasingly abstract internal representations from existing ones. This concept has been discussed in [3], for example. The use of supervised learning methods in order to infer information from such internal representations has been discussed in [4] and [5] and was applied to robot control and manipulation tasks [6] [7] [8]. The importance of self-organizing maps for the design and training of modular neural networks has been emphasized in [5].

The main contribution of the work presented in this paper is twofold: On the one hand, we want to demonstrate that new internal representations as well as mechanisms to infer knowledge from them can form concurrently and autonomously, without requiring explicit teaching. On the other hand, we want to demonstrate that the proposed combination of Hebbian learning and self-organizing maps is sufficiently robust to be applied in the context of a vision system dealing with unconstrained real-world environments such as traffic scenes. We consider this contrasting typical applications in robotics, where the environmental conditions are usually constrained and well-defined.

This paper is organized as follows. In section II, we briefly review the concept of cross-module learning [9] and the neural principles underlying the system architecture that we propose in section III. We then consider a simple case of correlation learning that involves three neural maps in order to demonstrate the validity of our approach. In section IV, we present a technical system instance of the proposed system architecture that is able to learn this correlation at run-time

M. Heracles is with the Research Institute for Cognition and Robotics (CoR-Lab) in Bielefeld, Germany. E-mail: heracles@cor-lab.uni-bielefeld.de

A. Gepperth, J. Fritsch and C. Goerick are with the Honda Research Institute Europe (HRI-EU) in Offenbach, Germany. E-mail: {alexander.gepperth, jannik.fritsch, christian.goerick}@honda-ri.de

and, after that, to predict either of the neural maps given the others. In order to evaluate the quality of these predictions, we have conducted experiments using both simulated and real-world image streams, which we describe and discuss in section V. Finally, we summarize the main contributions of this paper in section VI.

II. Basic concepts

In this section, we briefly review the concept of cross-module learning according to which the system architecture we propose in section IV is designed, as well as the underlying neural principles we use. The purpose is to provide a short overview of these concepts for the convenience of the reader. It is not meant as an exhaustive description, since these concepts are either well-known or have already been presented in detail in the given references. The focus of this paper is therefore not on the individual concepts but rather on the way we combine them into an integrated system architecture (see section IV).

A. Cross-module learning

Cross-module learning refers to a concept for the creation of complex large-scale systems that has been proposed in [9]. It emphasizes that such systems should be composed of individual components, in a way that facilitates the exchange of information between components throughout the entire system. This serves to enable system-wide learning and exploitation of interrelationships, or correlations, between various internal representations that the system has formed.

To achieve this, the concept of cross-module learning formulates a number of constraints that should be taken into account when designing large-scale systems. The most important are the use of a common data format (CDF) for the internal representations of the system which is shared by all components, and the use of generic learning and data fusion components that operate on these representations. Figure 1 shows an abstract example of a system architecture that satisfies these constraints.

The system architecture we propose in section IV is designed according to the principles of cross-module learning. In particular, we employ population-coded neural maps as common data format (see section II-B), dynamic neural fields in order to model their temporal dynamics (see section II-C), and combine Hebbian synaptic plasticity (see section II-D) with self-organizing neural maps (see section II-E) in order to achieve a generic learning mechanism. These techniques are briefly reviewed in the following.

B. Population-coded neural maps

Population coding [10] is a biological principle for encoding information in (two-dimensional) neural populations. Information is not encoded by the amplitude but rather by the location of neural activity in the population. Instead, the amplitude of neural activity can be interpreted as a confidence measure.

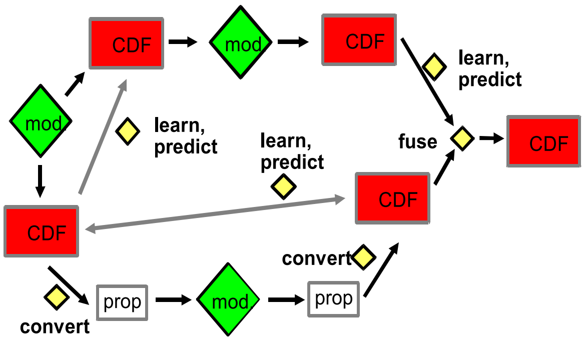

Fig. 1. Abstract example of a system architecture that is designed according to the principles of cross-module learning. The most important are the modularization of the system by the use of algorithmic components (large diamond boxes), the use of a common data format for their input and output data (large rectangular boxes), and the employment of generic components for learning and data fusion (small diamond boxes), thus achieving system-wide learning and exploitation of correlations.

For example, if a neural population encodes object categories, neural activity at different locations within this population represents different object categories. The amplitude of neural activity at a certain location represents the confidence that an object belongs to the object category corresponding to this location. Neural activity at more than one location, having similar amplitudes, represents that it is not clear which of the corresponding categories an object belongs to. This way, uncertainty can be expressed in population-coded neural maps, including the possibility to express that some hypotheses are more likely than others.

We employ population coded neural maps as a common data format for encoding all internal representations of the system, including sensory input data and intermediate representations.

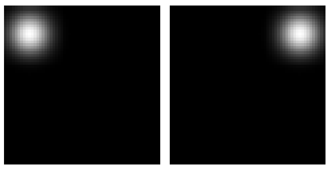

Fig. 2. Example of a topographical neural map representing a continuous scalar quantity that comes with a similarity measure. Left: The neural map representing the lowest value the quantity can assume. Right: The neural map representing the highest value the quantity can assume.

More specifically, continuous scalar values such as the 2D retinal size of objects or their 3D distance to the camera are encoded by a single Gaussian activation along the x-axis of a neural map (see figure 2). This mapping is topographic, i.e. similar values are encoded by Gaussian activations at similar positions in the map. In order to make use of the entire range of a neural map, the value to be encoded is scaled by the factor

$$\kappa = \frac{v_{\text{max}} - v_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

where v_{min} and v_{max} denote the minimal and maximal value to be encoded, respectively, and x_{min} and x_{max} denote the borders of the neural map. The location of the Gaussian activation with respect to the x-axis of the map is then obtained as

$$x = \kappa v$$

For scalar values, the location with respect to the y-axis can be set to any constant value since it does not encode any information. The scaling ensures that the value is encoded at the highest resolution possible, given the dimensions of the neural map, which is important if the neural map is to be correlated with another neural map by Hebbian learning. Note that each neural map has its own scaling factor. Regarding the opposite direction, given a neural map with maximum activity at location x^* with respect to the x-axis of the map, and the (known) scaling factor κ, the corresponding scalar value v^* can be obtained as

$$v^* = \frac{x^*}{\kappa}$$

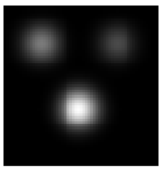

Fig. 3. Example of a non-topographical neural map representing a discrete scalar quantity that does not come with a similarity measure. Note that by design, the centers of activation are equi-distant from each other.

Discrete scalar values that do not come with a similarity measure, e.g. object class IDs as obtained by an object classifier, are encoded in non-topographic neural maps. Each value is encoded by a Gaussian activation at a pre-defined position in the neural map, all these positions being equi-distant from each other (see figure 3).

C. Dynamic neural fields

In order to model the temporal dynamics of the neural maps introduced in section II-B, we employ dynamic neural fields (DNF, see [11]). DNFs are a class of recurrent neural networks with fixed lateral connectivity. Basic units of this model are two-dimensional layers of rate-coded model neurons $u_{\vec{x},t}$ evolving according to the temporally discretized equation

$$\begin{aligned} u_{\vec{x},t+1} =& (1-\chi)u_{\vec{x},t} + \chi\{\alpha f_{\text{I}}[I_{\vec{x},t}] + \\ & \beta \sum_{\vec{x}'} w(\vec{x}' - \vec{x}) f_{\text{u}}[u_{\vec{x},t}] + u_{rest} + \gamma\sigma(t)\} \end{aligned} \tag{1}$$

$$\text{where } \chi \equiv \frac{\Delta t}{\tau}$$

$$f_{\text{u}}[z] = \{1 + \text{e}^{-\frac{2(z-\theta_u)}{\sigma_u}}\}^{-1}$$

$$f_{\text{I}}[z] = \text{atan}\frac{2(z-\theta_{\text{I}})}{\sigma_{\text{I}}}$$

Here, Δt and τ denote the time discretization step and the time constant of the model, respectively. The numbers α, β, γ are parameters of the model, $I_{\vec{x},t}$ denotes the sum of all inputs to the neural field, $\sigma(t)$ is Gaussian white noise and u_{rest} denotes the *resting potential* as a simple form of global inhibition. By the point-wise application of *transfer functions* $f_{\text{u}}[x]$ and $f_{\text{I}}[x]$, we regulate the input and output activity in the neural field. The lateral connectivity is defined by the *interaction kernel* $w(\vec{x} - \vec{x}')$, for which we use a Difference-of-Gaussians filter with variances σ^2_{on} and σ^2_{off}.

Average activity is controlled by a recurrent inhibition mechanism mediated by a single interneuron evolving according to equation (1) using a time constant τ_{FB}. These are bi-directionally connected to all model neurons of a layer. While the interneuron receives an activity average of the whole neural layer as input signal, each neuron in a layer receives the state of the interneuron, scaled by a constant w_{FB}. Since there are no lateral connections introduced by the interneuron and its afferent connectivity is fixed, the only additional free parameter introduced by this activity control mechanism is the time constant τ_{FB}. This mechanism is comparable to the proposal of [12].

By the "bubble solution" property of two-dimensional DNFs [11] it is ensured that neural map activities will converge to a population-coded state, provided that the inputs are population coded themselves. In particular, there is a class of solutions for DNFs characterized by a localized "bubble" of excitation [11]. This is important for the interaction of DNFs with the population coding approach chosen here (see section II-B). Details can be found in [13], [11], [14], for example. In the following, we will refer to the temporal dynamics of two-dimensional neural fields as *neural dynamics*.

D. Hebbian synaptic plasticity

The dynamic transmission of information between two neural maps $u_{\vec{x},t}$ and $v_{\vec{x},t}$, which is also referred to as *synaptic dynamics*, is governed by the equation

$$I^{v\rightarrow u}_{\vec{x},t+1} = \sum_{\vec{x}'} w^{vu}_{\vec{x}\vec{y},t} f_{\text{u}}[u_{\vec{x},t}] \tag{2}$$

where $w^{vu}_{\vec{x}\vec{y},t}$ represents the *weight*, i.e. the strength, of the model synapse between two neurons $u_{\vec{x},t}$ and $v_{\vec{y},t}$. The quantities $f_{\text{u}}[z]$, $f_{\text{I}}[z]$ and $I^{v\rightarrow u}_{\vec{x},t+1}$ are the same as defined in section II-C. In parallel, weight adaptation is performed according to the Hebbian learning rule

$$w^{vu}_{\vec{x}\vec{y},t+1} = w^{vu}_{\vec{x}\vec{y},t} + \epsilon_l f_{\text{u}}[v_{\vec{y},t}] f_{\text{u}}[u_{\vec{x},t}] - \epsilon_f w^{vu}_{\vec{x}\vec{y},t} \tag{3}$$

where ϵ_l and ϵ_f denote the learning and forgetting rate of the model, respectively. For maximum input, this mechanism has a stable fixed point for $w^{vu}_{\vec{x}\vec{y}} = \frac{\epsilon_l}{\epsilon_f}$, thus limiting weight growth.

While Hebbian synaptic plasticity provides a generic mechanism for learning correlations between two neural maps, it cannot directly be used to learn correlations between multiple neural maps. In order to overcome this limitation, we propose to combine Hebbian learning with self-organizing neural maps.

E. Self-organizing maps

Self-organizing maps (SOMs, see [2]) are a class of single-layered feed-forward neural networks which essentially perform a clustering of their input using an online learning rule.

Let $u_{\vec{x},t}$ denote the state of model neurons at position $\vec{x}$ and time t. Furthermore, let $\vec{v}_{\vec{x},t}$ denote the afferent weight vector attached to each neuron. In the presence of an input pattern $\vec{p}_t$ being fed into the network, let $\vec{x}^*$ denote the position of the best-matching unit (BMU), i.e. the neuron whose attached weight vector best fits the input. As for the SOM algorithm, let $g^{\sigma}_{\vec{x}-\vec{x}^*}$ denote a Gaussian neighborhood function centered at $\vec{x}^*$ with variance σ, and let ϵ_t denote the learning rate at time t. In these terms, let us reconsider the standard SOM algorithm as given in [2] in a slightly different form:

1) Initialize $\epsilon = \epsilon_0, \ \sigma = \sigma_0$
2) At time t, present pattern $\vec{p}_t$ and propagate activation to the model neurons according to $u_{\vec{x},t} = \vec{v}_{\vec{x},t} \cdot \vec{p}_t$.
3) Determine the BMU $n_{\vec{x}^*,t}$.
4) Adapt weight vector $\vec{v}_{\vec{x}^*}$:

$$\vec{v}_{\vec{x},t+1} = \vec{v}_{\vec{x},t} + \epsilon g^{\sigma}_{\vec{x}-\vec{x}^*} \{\vec{v}_{\vec{x},t} - \vec{p}_t\}. \tag{4}$$

5) Increase t by 1, calculate new values of ϵ and σ and continue at step 2.

As stated in [2], the temporal dynamics of the parameters ϵ and σ have a strong influence on the output layer of the SOM. In the work presented here, we use a parametric model, given as

$$\sigma(t) = \sigma_0 \frac{1}{\tau_\sigma t} \tag{5}$$

$$\epsilon(t) = \epsilon_0 \frac{1}{\tau_\epsilon t} \tag{6}$$

III. System architecture

Figure 4 depicts the basic structure of the system architecture we propose. It is compliant with the concept of cross-module learning (see section II-A) and employs the neural mechanisms that were briefly reviewed in section II. The most important aspects comprise

- the use of image processing methods in order to detect regions of interest (ROIs) in the image, localizing potential objects (e.g. cars),
- the extraction of object-specific features (e.g. 2D size and 3D distance) and their encoding in population-coded neural maps,
- the use of a self-organizing map for the unsupervised formation of an intermediary, higher-order neural map that represents a combination of the (lower-level) input maps, and
- the use of standard Hebbian learning in order to learn correlations between the higher-order map and another neural map (which can be a lower-level map like in the figure, or another higher-order map).

The crucial point is the mapping of multiple lower-level maps onto a single higher-order map representing combinations of the lower-level maps. This way, learning correlations between multiple neural maps can be reduced to learning correlations between two (higher-level) neural maps, which makes Hebbian learning applicable again. We consider the proposed combination of Hebbian learning with self-organizing maps a generic mechanism for system-wide unsupervised correlation learning, even in the case of multiple neural maps, which cannot be handled directly by standard Hebbian learning.

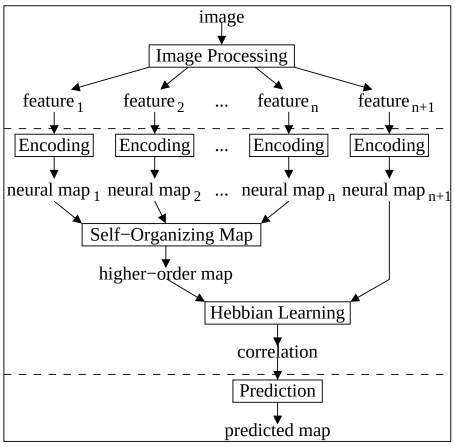

Fig. 4. Basic structure of the proposed system architecture, combining Hebbian learning with self-organizing maps. See section III for an explanation.

IV. Technical system instance

Before presenting the technical system instance of the system architecture we have proposed in section III, we briefly introduce the most important algorithmic components we use.

A. Algorithmic components

This section intends to give a brief overview over the most prominent algorithms used in the technical system instance described in section IV. Since the precise working of many algorithms is quite complex, we only give a general picture here and refer the interested reader to the cited references. We feel that this is a justified approach since, although the algorithms are necessary to achieve the technical system instance, they are not the focus of this publication.

1) Adaptive saliency maps for point-of-interest detection: In order to analyze a visual image for candidate objects, it is usually assumed that those objects can at least partly be distinguished from the background by certain visual properties. Thus, points of interest (POIs), which may be subject to further analysis, can be generated where such properties are present in the image. Motivated by studies of visual attention in humans [15], the concept of saliency maps as first proposed by [16] obtains POIs by analyzing several simple features computed from an image. In this contribution, we use the saliency map model proposed in [17] which extends previous models by considering more sophisticated image features, and the ability to adapt its processing parameters to detect POIs in different situations. Output of the model is an activity-encoded image (a *saliency map*), of which we take the point of maximal activation to be the POI for subsequent processing stages.

2) Image segmentation for region-of-interest generation: For extracting regions of interest (ROIs) around a POI, we use a simple color-based region-growing approach [18], with the seed value computed at the POI. For the described experiments, this method was able to compute size estimates of sufficient accuracy to allow the prediction of distance. It is nevertheless clear that for complex scenes, a more sophisticated method will have to be employed in order to obtain precise and informative estimates of object size.

3) Object recognition by a visual hierarchy: We use the Visual Hierarchy object recognition system as described in [19] to determine the identity of an ROI's content. The recognition system (also termed "classifier") is designed to be trained prior to application using training examples of known identity. It is essentially a feed-forward neural network, enriched by successive pooling and competitive and self-organizing stages, which models the convergent hierarchical processing in the human visual system. As will be described in section V, the classifier is used here for discriminating real-world objects such as cars from the background, as well as artificial objects (toy duck, cola can) in a simulated environment. The training procedure and the adaptation of the classifier architecture to the car domain using color information has been described in [20]. For artificial objects, the procedure of [20] is used, however with a strongly reduced number of training examples.

4) Stereo vision for 3D distance estimation: Distance estimates are obtained using the commercially available SVS stereo-correlation software [21], which operates on a pair of images provided by a stereo camera. The output is a disparity image. Since the stereo camera is calibrated, the disparity image can be transformed into a 3D distance map by using the camera parameters. The SVS algorithm implements a local window-based correlation method, therefore it suffers from the aperture problem which leads to unreliable disparity and 3D distance estimates. This is a problem that is inherent in any local correlation method, including optical flow approaches. However, the SVS algorithm also provides a confidence value for each pixel in the disparity image. In practice, all disparities in the disparity image are marked as invalid if their confidence is below a certain threshold. This leads to a disparity image in which some regions have dense disparities, while other regions may not have valid disparities at all.

5) Implementation issues: For all implementation purposes, we use the component-based middle-ware system RTBOS [22] which allows the creation of large applications with an emphasis on scalability (automatic distribution over several machines) and computational efficiency (all components written in C/C++). All functionalities described in this contribution are realized as RTBOS components. In this way, a favorable processing speed (10 frames per second on two off-the-shelf notebook computers) can be achieved while the details of the parallelization are decoupled from the application development process.

B. Real-time implementation

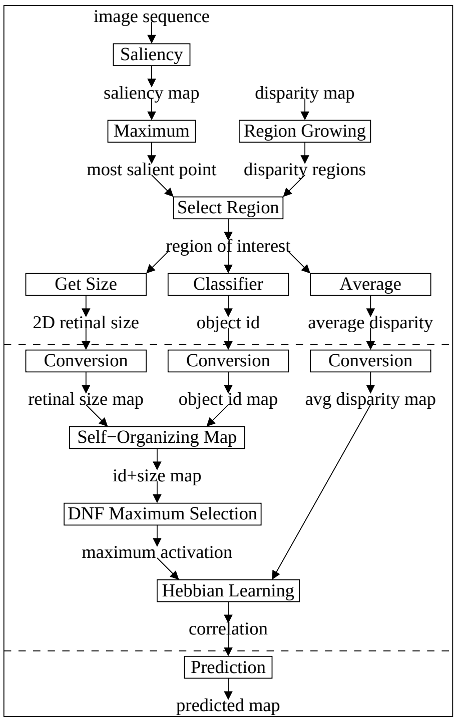

Fig. 5. Detailed overview of the proposed system architecture. See section IV for a detailed explanation.

The real-time implementation (shown in detail in figure 5) operates on a sequence of camera images which is processed frame by frame as follows: As a first step, a saliency map is computed, using the method described in section IV-A.1. From the saliency map, the most salient point in the image is determined. A region growing approach (see section IV-A.2) operating on a disparity map (see section IV-A.4) corresponding to the camera image then determines an area of similar disparities, where the most salient point is used as seed point. The ROI thus obtained is represented as a bounding box in the image. Given this ROI, different quantities are computed from its content and each quantity is represented in an individual neural map. The quantities comprise the average 3D distance of all pixels in the ROI (in terms of disparity), the 2D retinal size calculated as the area of the bounding box, and the object class ID obtained as the result of the object classifier operating on the ROI. As for the latter, we use the object classifier described in section IV-A.3. It is important to note that we assume that this object classifier has been trained to recognize the relevant objects in the scene beforehand, as described in IV-A.3. The conversion between the resulting scalar values of the quantities and the neural maps representing them is performed as described in section II-B. The resulting population-coded neural maps representing the 2D object size and the object class are then fed into a SOM as afferent inputs, which adapts the weights between the afferent inputs and its output activation layer according to section II-E. Thereby, pairs of object class and 2D object size are mapped onto a single neural map, which is the intermediary, higher-order neural map mentioned before. Thanks to the properties of self-organizing maps, this higher-order neural map is again topographical, i.e. similar combinations of object class and 2D object size are represented at similar positions. The output layer of the SOM is connected to the neural map encoding the average 3D distance of the pixels in the bounding box, adapting the connection weights by a Hebbian (correlation-based) learning mechanism (see section II-D). This way, the system is able to detect correlations and dependencies between these two neural maps. Essentially, dependencies between the higher-order map representing a combination of 2D object size and object class on the one hand and the (simple) neural map representing the average 3D distance of the object on the other hand can now be learned by direct association.

V. Experiments

From the learned correlation between the higher-order map representing combinations of the 2D size and the class of an object on the one hand and the neural map representing its average 3D distance on the other hand (see section IV), the system should be able to predict either of the two given the other. For example, if the system perceives an object and recognizes its object class, it should be able to predict its 3D distance from its 2D retinal size alone. We have conducted two experiments in order to evaluate the quality of these predictions.

A. *Experimental setup*

The real-time implementation of the proposed system architecture is executed in an offline setting, operating on a recorded image sequence instead of a live-stream. In the first experiment, the image sequence has been recorded from a simulated 3D environment in which two objects at camera height were randomly moved back and forth, independently of each other. The purpose of this experiment is to establish the feasibility of the chosen approach under ideal conditions, as well as to demonstrate that the mechanism is capable of handling distinct object classes with different size-distance dependencies. The distance of each object with respect to the camera varied between 0.75 m and 2.00 m. In the second experiment, the image sequence has been recorded from a real-world traffic scene in which a car waiting at a red traffic light is approached. This experiment is conducted in order to demonstrate that our approach is capable of dealing with the complexity of real-world scenes. Example images of both streams are shown in figures 6 and 7, respectively. In both experiments, the image streams are repeated after the last frame in order to provide a continuous input.

For both experiments, we assume that the object classifier has been trained beforehand to recognize the objects in the scene (see section IV-A.3). The first 2000 frames of the image sequence are processed by our system without evaluation. This serves as a training phase, giving the learning methods

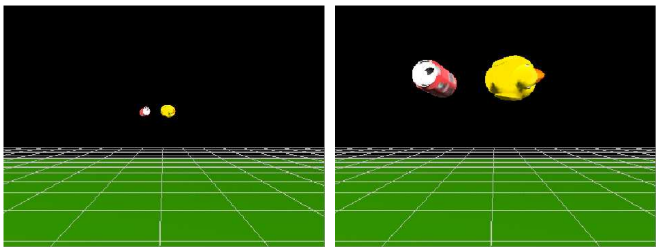
Fig. 6. Example images of the simulator sequence used in the first experiment. Left: Both objects are as far away as possible. Right: Both objects are as close as possible. Note that the distance of the objects is varied independently of each other.

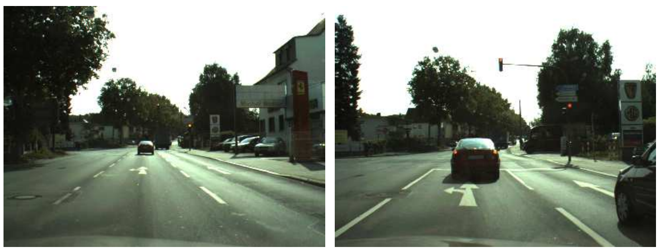
Fig. 7. Example images of the real-world sequence used in the second experiment. Left: The car being far away. Right: The car being close.

enough time to learn the aforementioned correlation. As soon as the first 2000 frames have been processed, learning is stopped and evaluation is being enabled. Table I provides an overview of the parameters used in the experiment.

TABLE I

PARAMETER VALUES USED IN THE EXPERIMENT

Method	Parameters	Values
SOM	r_0, ϵ_0	5.0, 1.0
Hebbian learning	ϵ_l, ϵ_f	0.001, 0.001
Neural field	$\tau, \alpha, \beta, \gamma, u_{\text{rest}}$	10,1.0, 5.0, 0.1,-0.4

B. *Evaluation measures*

For the sake of a simple yet systematic evaluation, the prediction accuracy is evaluated for one of the two objects only in the first experiment (once the training phase is over). The results are unchanged if the other object is used for evaluation. There is no such change in the second experiment since the car is the only object that can sensibly be focussed in the used video stream. For each image frame i, we compare the actual 3D distance of the object as obtained by stereo vision (see section IV-A.4) with the 3D distance of the object as predicted by our system after the training phase. The latter is determined by finding the maximum activation in the neural prediction map (see bottom of Fig. 5) and translating it back into a real number by the procedure explained in section II-B. Let δ_i denote the difference between the actual and the predicted 3D distance of the object. We define the following measure in order to evaluate the quality of the prediction. The measure is defined as the root of the squared prediction error, averaged over the image frames

$i \in \{1, \ldots, N\}$ after the training phase:

$$\delta_{\text{RMSE}} \equiv \sqrt{\frac{1}{N} \sum_i \delta_i^2}. \qquad (7)$$

C. Results

The results of the two experiments are depicted in figures 8 and 9, respectively. It is worth noting that the complete system instance used for both conducted experiments runs in real-time (10+ frames per second) on a standard desktop computer. In terms of the (scaled) disparity, the dark green graph shows the 3D distance of the focused object as predicted by our system while the light green graph shows the 3D distance of the object as obtained by stereo vision, using the SVS algorithm (see section IV-A.4). The light red graph shows the prediction error in terms of the absolute difference between the two green graphs. The dark red graph represents the average prediction error.

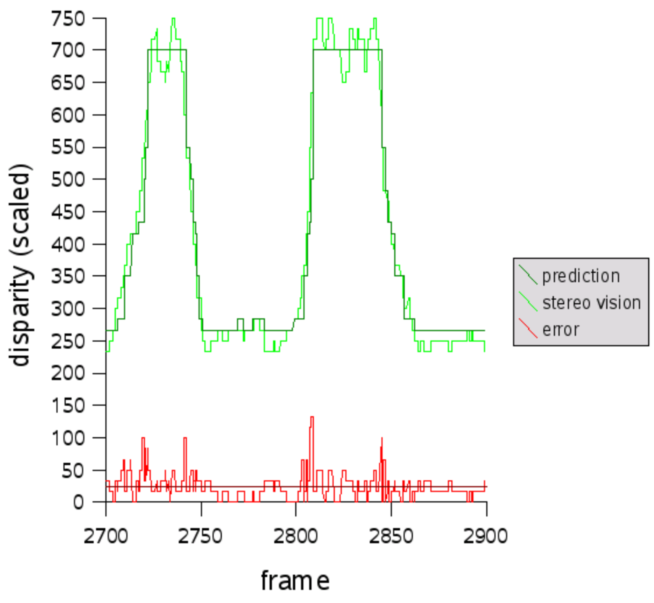

Fig. 8. Results of the first experiment (simulated sequence).

D. Discussion

Obviously, the presented system is able to predict the 3D distance of visual objects. This is true for the entire range of distances that occur, as can be seen in frames 2700 – 2725 in figure 8 and in frames 2000 – 2025 in figure 9, for example. In both cases, the focused object is far away at the beginning (corresponding to low disparity values) and then gets increasingly closer (corresponding to high disparity values). A closer look at the error graphs reveals that the prediction error does not seem to depend significantly on the distance itself, instead, the error is rather the same for the object being close or far away. However, one can see by comparing figures 8 and 9 that the prediction is significantly coarser for the real-world sequence than it is for the simulated sequence. We attribute this to the fact that in the real-world sequence, it is more difficult to reliably

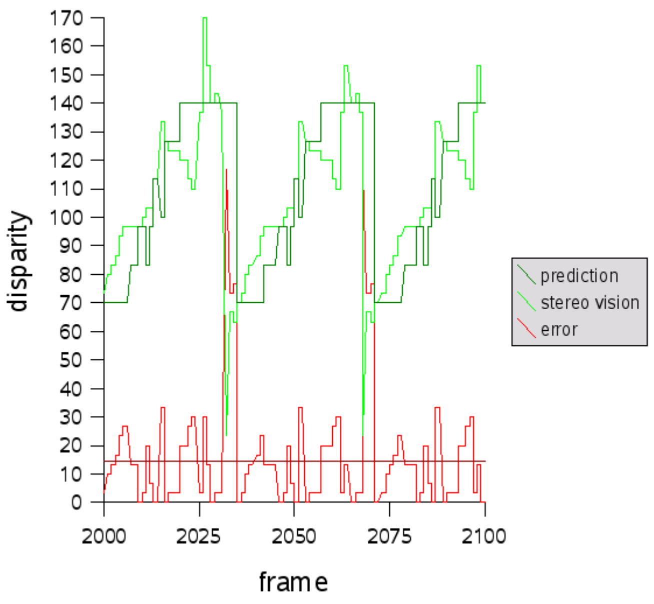

Fig. 9. Results of the second experiment (real-world sequence).

segment the focused object, which leads to inaccuracies in the ROI sizes that are involved in learning the correlation. We argue that the use of a more sophisticated segmentation component would compensate for this effect and increase the accuracy of the prediction.

At the conceptual level, we have shown that the combination of self-organization, nonlinear maximum selection and associative learning can produce a new quality of predictions. We have demonstrated that individual, strongly non-linear prediction models can be acquired autonomously for different object classes. Moreover, the described technical system instance is capable of real-time operation and therefore well suited for application in, .e.g, the car domain. Extending previous work, our model introduces a strong nonlinearity (the maximum selection, in contrast to [5], [3]) on the one hand, and a experimental evaluation on visual sensory data on the other hand (as opposed to [6], [7]).

VI. Conclusion

We have presented an architecture that enables a system to learn correlations between multiple neural maps, which can then be used to make predictions about one map given the others. This is achieved by the self-organized formation of an intermediary, higher-order map that combines multiple neural map representations. The correlations can then be learned by means of standard Hebbian learning methods. Both the self-organizing maps and the Hebbian learning can evolve in parallel during run-time of the system. In our experiments we have shown that the proposed architecture can, for example, learn the correlation between the 2D retinal size of objects, the object class they belong to and their 3D distance from the viewer, and that it can then use the learned correlation to predict the 3D distance of objects from their 2D retinal size alone, provided that it recognizes the object class. Since the proposed architecture represents a way to learn correlations between multiple neural maps, we consider the aforementioned combination of self-organizing maps and Hebbian learning a generic principle for learning correlations at various levels within large-scale systems.

REFERENCES

[1] D Hebb. *The organization of behaviour*. Wiley, New York, 1949.

[2] T Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernet.*, 43:59–69, 1982.

[3] H Ritter. Self-organizing maps for internal representations. *Psychological Research*, 52(2-3), 1990.

[4] J Walter, H Ritter, and K Schulten. Non-linear prediction with self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks*, pages 589–594, 1990.

[5] J-L Buessler, J-P Urban, and J Gresser. Additive composition of supervised self-organizing maps. *Neural Processing Letters*, pages 9–20, 2002.

[6] G Barreto, A Araújo, and H Ritter. Self-organizing feature maps for modeling and control of robotic manipulators. *Journal of Intelligent and Robotic Systems*, 2003.

[7] J-L Buessler, D Kuhn, and JP Urban. Learning self-organizing maps using input-output associations applied to robotics. In *Proceedings of the World Congress on Neural Networks*, volume 2, pages 384–388, 1995.

[8] B Ridge, D Skocaj, and A Leonardis. A system for learning basic object affordances using a self-organizing map. In T Asfour and R Dillmann, editors, *Proceedings of the First International Conference on Cognitive Systems*, 2008.

[9] A Gepperth, J Fritsch, and C Goerick. Cross-module learning as a first step towards a cognitive system concept. In *Proceedings of the First International Conference On Cognitive Systems*, 2008.

[10] RS Zemel, P Dayan, and P Pouget. Probabilistic interpretation of population codes. *Neural Computation*, 10(2):403–430, Feb 1998.

[11] JG Taylor. Neural 'bubble' dynamics in two dimensions: foundations. *Biological Cybernetics*, 80:393–409, 1999.

[12] C Wilimzig, S Schneider, and G Schöner. The time course of saccadic decision making: dynamic field theory. *Neural Networks*, 19(8):1059–1074, Oct 2006.

[13] S-I Amari. Mathematical foundations of neurocomputing. *Proceedings of the IEEE*, 78(9):1441–1463, 1990.

[14] Inna Mikhailova and Christian Goerick. Conditions of activity bubble uniqueness in dynamic neural fields. *Biological Cybernetics*, 92(2):82–91, Feb 2005.

[15] R Desimone and J Duncan. Neural mechanisms of selective visual attention. *Annual Reviews Neuroscience*, 18:193–222, 1995.

[16] L. Itti and C. Koch. Computational modeling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, Mar 2001.

[17] T Michalke, J Fritsch, and C Goerick. Enhancing robustness of a saliency-based attention system for driver assistance. In *The 6th International Conference on Computer Vision Systems (ICVS)*, Lecture Notes in Computer Science (LNCS), Santorini, Greece, 2008. Springer Verlag Berlin Heidelberg New York.

[18] B Jähne. *Digital image processing*. Springer Verlag Berlin, Heidelberg, New York, 6 edition, 2005.

[19] H Wersing and E Körner. Learning optimized features for hierarchical models of invariant object recognition. *Neural Computation*, 15(7), 2003.

[20] A Gepperth, B Mersch, J Fritsch, and C Goerick. Color object recognition in real-world scenes. In JM de Sa, editor, *ICANN 2007, part II*, number 4669 in Lecture Notes in Computer Science. Springer Verlag Berlin Heidelberg New York, 2007.

[21] K Konolige. Small vision system: hardware and implementation. In *8th international symposium on Robotics Research, Japan*, 1997.

[22] A Ceravola, M Stein, and C Goerick. Researching and developing a real-time infrastructure for intelligent systems. *Robotics and Autonomous Systems*, 56(1):14–28, 2007.