Associative Language Processing in Cortical Areas

Heiner Markert, Andreas Knoblauch, Günther Palm

2005

Preprint:

This is an accepted article published in Proceedings of the IEEE SMC UK-RI Chapter Conference 2005 on Applied Cybernetics. The final authenticated version is available online at: https://doi.org/[DOI not available]

Associative Language Processing in Cortical Areas

Heiner Markert[1], Andreas Knoblauch[2] and Günther Palm[1]
[1]Abteilung Neuroinformatik, Fakultät für Informatik, Universität Ulm,
Oberer Eselsberg, D-89069 Ulm, Germany
Tel: (+49)-731-50-24151; Fax: (+49)-731-50-24156
{markert,palm}@neuro.informatik.uni-ulm.de
[2]Honda Research Institute Europe GmbH
Carl-Legien-Str. 30
D-63073 Offenbach/Main, Germany
Tel: (+49)-69-89011-761; Fax: (+49)-69-89011-749
andreas.knoblauch@honda-ri.de

Abstract − *We have implemented a system that can learn to associate words with objects, properties like colours and form as well as actions. The model is based on associative memories using sparse distributed representations. The system is used in a robotics scenario where a robot has to respond to spoken or typed commands like "bot show plum" or "this is cup". This involves parsing and understanding of simple sentences and relating nouns to concrete objects sensed by the camera. The model is able to learn new objects at run time.*

Keywords: associative memory, sparse patterns, distributed representation, neural assemblies, Hebbian learning, language understanding, global brain modelling

1 Introduction

When humans are processing language referring to actions or visual scenes, distributed cortical networks including motor areas and parts of the visual systems become active [1]. There appear to be strongly coupled neuron ensembles in the brain, correlating between words and their referred actions and objects. The theory of cell assemblies [2–5] provides one of the most promising frameworks for modelling and understanding the brain in terms of distributed neuronal activity. It is suggested that entities of the outside world as well as internal states are coded in groups of neurons and that a neural cell assembly is generated by Hebbian coincidence learning [6] where the synaptic connections are strengthened between co-activated neurons. Thus models of neural associative memory have been developed as abstract models for cell assemblies.

In this work we describe a neurobiologically plausible model of language processing based on cell assemblies [2–4]. We have developed a system that can learn to associate words with objects, properties like colours, and actions. This system is used in a robotics context to enable a robot to respond to spoken or typed commands like "bot show plum" or "bot put apple to yellow cup". This involves parsing and understanding of simple sentences and relating nouns to concrete objects sensed by the camera and recognised by a neural network from the visual input.

2 Neural associative memory

We decided to use the Willshaw associative memory [7, 8, 3] as a single framework for the implementation of cell assemblies in cortical areas. The idea of cell assemblies goes back to Hebb [2], and we have chosen the Willshaw model mainly because it is a biological plausible while still simple implementation of the idea of cell assemblies. The Willshaw model seems more realistic than e.g. the Hopfield model [9] as in the latter single synapses need to change from inhibitory to excitatory behaviour during learning, which does not happen in biological systems. It is also the most effective associative memory mechanism in terms of storage capacity and information efficiency [10–13].

A cortical area consists of n binary neurons which are completely connected by binary synapses. A cell assembly or pattern is a binary vector of length n where k one-entries in the vector correspond to the neurons belonging to the assembly. Usually k is much smaller than n. Assemblies are represented autoassociatively in the synaptic connectivity such that any two neurons of an assembly are bidirectionally connected.

Instead of classical one-step retrieval we use the spike counter model, an improved architecture based on spiking associative memory [14]. The model is explained in detail in section 3. A cortical area is modelled as a local population of n neurons which receive input from other areas via Hebbian learnt heteroassociative connections. The model is simulated in global time steps with relatively low temporal resolution. Within each global step, each area computes exactly one pattern retrieval which requires only local information and therefore can be simulated with higher temporal resolution. Basically, the neurons receiving the strongest heteroassociative external input will fire first, and all emitted spikes are fed back immediately through Hebbian learnt auto-associative connections. Depending on the so called threshold parameter α of the model, this can lead to activation of single assemblies as well as activating all assemblies similar to the address pattern. In comparison to the classical model, this model has a number of additional advantages. For example, assemblies of different size k can be stored and input superpositions of several assemblies can more easily be separated. Even more, the model allows for automatic detection of the address quality. This could for example be used to adjust the threshold parameter according to the input quality. The systems behaviour will then change depending on the address quality: If there is an assembly which is addressed more "consistent" than all others, this and only this assembly will be activated. If a decision for one single assembly is not possible, the threshold will be decreased leading to the parallel activation of similar assemblies, representing uncertainty or ambiguity in that specific area.

3 The spike counter model

This section will describe the spike counter model, which is the underlying principle of the whole system, in greater detail.

Let N be the number of neurons in the whole network. Our architecture divides the network in to several disjoint areas, which we will call $P_1, P_2, \ldots, P_M$. Note that each neuron must belong to exactly one population P_i. For simpler notation, we also enumerate the neurons according to the population they belong to, e.g. the first $|P_1|$ neurons belong to population P_1, the neurons $|P_1|+1, \ldots, |P_1|+|P_2|$ belong to P_2 and so on.

There is an autoassociative connection matrix A_{P_i} for each population P_i where the synaptic weights of the autoassociative feedback of the area onto itself are stored. Furthermore, any two populations P_i and P_j can be heteroassociatively connected to each other via a coupling matrix $H^k_{P_iP_j}$. A non-negative delay matrix $D^k_{P_iP_j}$ is assigned to each heteroassociation. The entries of the delay matrix are integral numbers and give the delay in global time steps (see below). The index k allows for several heteroassociative connections between the same populations, e.g. with different delay values.

The matrices A_{P_i} can be written into one large autoassociative feedback matrix for the whole system by putting each A_{P_i} on the i-th position of the diagonal and zeros elsewhere. Similarly, heteroassociative matrices H^k with corresponding delay matrices D^k are constructed from the sets $(H^k_{P_iP_j})_{ij}$ and $(D^k_{P_iP_j})_{ij}$ (put $H^k_{P_iP_j}$ at position (i,j) of the matrix H^k).

Before the above notation can be used to formulate the differential equations the system is based on, we need to introduce a special notation of time. For efficiency reasons, the model uses two different time scales, a global discrete time s and during each global time step a relative continuous time t. Let the global time steps be $s_1, s_2, \ldots$. Then, in each global time step the following operations are performed:

- All output values are fixed in the whole system and each population is performing exactly one pattern retrieval. As the retrieval itself is a dynamical process, we introduce a relative time t^{s_i} in each global time step s_i, which will be used to calculate spike times of the neurons relative to the global time step.
- If all retrievals are finished, we propagate the new spike activity through the heteroassociative connections.

Let x denote the membrane potential of one specific neuron (with some index $i \in \{1, \ldots, N\}$ which we do not always write down to keep the equations simpler). Then, in global time step s, the model is given by $x_s(0) = 0$ and

$$\dot{x}_s(t^s) = a_1 \cdot I\left(c^H_{s-1}, c^F_{s-1}\right) + a_2 \cdot F\left(c^H_{s-1}, c^F_{s-1}, c^A_s, c^\Sigma_s\right), \tag{1}$$

where a spike is emitted as soon as x exceeds threshold Θ. Here, I is the initialisation function which only depends on the heteroassociative input and autoassociative feedback input and is constant over the whole time step s. F is called input integration function and is the most important term characterising the retrieval dynamics. The values a_i are simply real valued weighting factors. The c^X_s-parameters are real-valued functions of t which describe different state aspects of the system at a given global time step s and relative time t. If t is infinite, the value usually refers to the complete global time step s.

Before we give a detailed description of I and F below in equations (10) and (11), we first introduce the idea of spike counters. Basically, the parameters c^H, c^F, c^A and c^Σ (we dropped the index s here for simplicity) all count several spikes that happened in a specific time step until a given time. What makes

them different is the set of neurons they count spikes on and the weight a spike gets. In more detail, c^H is counting the number of heteroassociative input spikes a neuron gets, i.e. if we look at $c^H_{s-1}(\infty)$ in time step s for a neuron with index i, the counter value is the number of neurons that emitted a spike in time step $s-1$ and that are connected heteroassociatively to neuron i. Additionally, c^H is a weighted sum where early spikes are more important. Similarly, c^F counts the number of autoassociative feedback spikes, i.e. the number of spikes that a neuron received via its autoassociative feedback matrix from the previous time step, weighted in the same way as c^H. These two are mainly used for initialising the retrieval, as they correspond to the external input of a population. The parameter c^A is used during retrieval. For each neuron i in a time step s at relative time t, it gives the number of neurons that already spiked in time step s before time t and that are connected to neuron i autoassociatively. The counter c^Σ is similar to c^A, but instead of counting only the neurons that have a connection to a specific neuron i, it counts all spikes that occurred in the population of neuron i at time step s before time t, i.e. c^Σ gives a measure of the total activity in the population. Both c^A and c^Σ are non-weighted sums over the active neurons.

Formally, the spike counters are given by

$$(c^H_s)_j(t) = \sum_k \sum_i \frac{\Theta}{\left(T(s-D^k_{ij},t)\right)_i} \cdot H^k_{ij} \quad (2)$$

$$(c^F_s)_j(t) = \sum_i \frac{\Theta}{(T(s,t))_i} \cdot A_{ij} \quad (3)$$

$$(c^A_s)_j(t) = \sum_i y_i(t) \cdot A_{ij} \quad (4)$$

$$(c^\Sigma_s)_j(t) = \sum_{i \in P(j)} y_i(t) \quad (5)$$

where $P(j)$ is the population containing the neuron with index j. In the above equations, we made use of the state variables

$$y^s_i(t) = \mathbf{1}_{\{x^s_i(t) \geq \Theta\}}(t) \quad (6)$$

$$(T(s,t_{\max}))_i = \min\{t \leq t_{\max} : y^s_i(t) = 1\}. \quad (7)$$

The value y is 1 if the neuron i already emitted a spike in global time step s before time t, zero otherwise. The value T is derived from y and is the time of the first spike of neuron i in the global time step s before time $t_{\max}$. In (7), the minimum over an empty set is ∞, i.e. $(T(s,t_{\max}))_i = \infty$ if $\{t \leq t_{\max} : y^s_i(t) = 1\} = \emptyset$.

We now define two helper functions:

$$h^s_w(t) = c^H_{s-1}(t) + w \cdot c^F_{s-1}(t) \quad (8)$$

$$k^s(t) = c^H_{s-1}(\infty) \cdot \left(\frac{c^A_s(t)}{c^\Sigma_s(t)} \cdot (a_6 - a_5) + a_5 \right), \quad (9)$$

where $k^s(t) = 0$ if $c^\Sigma_s(t) = 0$. Finally, we can define the initialisation function I and the input integration function F as follows:

$$I(c^H_{s-1}, c^F_{s-1}) = h^s_{a_3}(\infty) \quad (10)$$

$$F(c^H_{s-1}, c^F_{s-1}, c^A_s, c^\Sigma_s) = h^s_{a_4}(t) + \alpha \cdot k^s(t) \quad (11)$$

Here, $\alpha \in [0,1]$ and a_3 to a_6 are real-valued weight or scale factors. For our simulations, a_6 has a small positive value (typically 0.02), while a_5 is around -50. The values of a_3 and a_4 control how strong the autoassociative feedback of a population is and depend on the population under consideration. Typical values for both a_3 and a_4 are in the range of zero to 100, where zero means no feedback at all, while 100 means very strong feedback.

Note that at a given time step s there is a finite relative time t^s where no more neurons are able to spike, as we allow every neuron to spike at most twice.

The input integration function (11) plays a major role during retrieval. It calculates some sort of confidence measure for each neuron whether it belongs to the addressed assembly (F positive) or not (F small or negative). If F is negative with high absolute value, $\dot{x}$ in equation (1) will become negative and the neuron will not reach threshold Θ at all, so it will not spike. If F is positive with high absolute value, $\dot{x}$ will be very large and the neuron will fire very soon. F basically is high, if c^A is about as high as c^Σ for a neuron and negative if $c^A << c^\Sigma$. The parameter α determines how fast F decreases with decreasing ratio of c^A to c^Σ, which in turn has an influence on the separation strength of the population: a lower α allows multiple assemblies to spike during one retrieval, while high values of α allow only one assembly to become active.

As we demonstrate in the following sections our model can be used to adequately process ambiguous inputs and, if necessary, learn new representations. For this it is necessary to recognise "bad" retrievals. By allowing the neurons to spike twice in each retrieval (reset x to zero after a spike) we determine the retrieval quality as follows: If exactly one assembly was addressed, it will spike twice, very early and almost synchronously with no other neurons spiking in between the two spikes. If the input quality becomes worse (e.g., when addressing with a superposition or noise), more neurons not belonging to one assembly will fire before the earliest second spikes appear. Also, only the overlap of the addressed patterns will belong to the earliest second spikes and not a whole assembly. By exploiting this information the basic confidence measure can be calculated.

Additionally, a simple retrieval with high α can be done for control purposes and allows for a larger variance in assembly size. The control retrieval can be used to determine the assembly with strongest input. Instead of counting the number of neurons which participated in the first spikes and comparing this value to the optimal assembly size, the control retrieval can

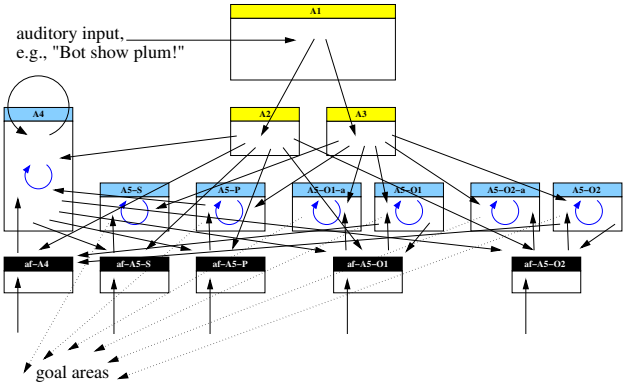

Figure 1: The language system consisting of 10 cortical areas (large boxes) and 5 thalamic activation fields (small boxes at bottom). Straight arrows correspond to inter-areal connections, while circular arrows correspond to short term memory.

be compared against the first spikes in the main retrieval.

Our basic model has a nice correspondence with well known neurobiological mechanisms which have been studied previously in a simpler but biologically more realistic model variant [14]. In particular, the spike counters can be interpreted as excitatory (c^A, c^H, c^F) and inhibitory (c^Σ) synaptic conductances, and the different weighting of early and late spikes corresponds to a spike-latency code relative to an underlying oscillation.

4 Cortical language model

Figure 1 gives an overview of our model for cortical language processing. The model mainly consists of 10 cortical areas, each of which is implemented using the spike counter model described in section 3. The system can understand simple sentences like "bot show red plum" or "bot lift apple". The combination of a sequence detector unit (called A4) used to store grammatical information and additional areas (A5-X) for storing the grammatical function of previously processed input words allow the model to parse regular grammars. For a more detailed description, see [15, 16].

The system is able to detect and correct ambiguous input on the single word level as long as enough context information is or becomes available (see also [15]). For example, the sentence "bot lift red bwall" with ambiguous input between "ball" and "wall" will be correctly interpreted as "bot lift red ball" by this model, because a wall is not a liftable object. Similarly, the sentence "bot show/lift green wall", with an artificial ambiguity between "show" and "lift", is correctly understood as "bot show green wall". This shows that even if the disambiguating context information arrives after the ambiguous input and even after an intermittent word ("green"), the system is still able to disambiguate the input. This works by controlling the threshold parameter of each single area by its retrieval quality as described in section 2.

5 Online Learning

The estimation of the retrieval quality enables the model to decide whether a presented pattern is already known to the system or if it is something completely new. We will show that this can be used to learn new objects online. The scenario for this is a robot close to one or two tables on which there are certain kinds of fruits and/or other simple objects [15]. The robot is given spoken or typed commands like "bot show red plum", on behalf of which it will have to search its vicinity for a red plum and point to it, if it found one. A task like this also requires attention control and object detection mechanisms in the system, for further details on that see e.g. [15].

Learning of new patterns will be initiated by a command like "this is cup". In a simplified scenario, giving this command requires that only one object is currently in the robots visual field, in a more complex scenario, the robot will have to look for gestures pointing to an object. The language processing system then realises the command "this is" and prepares the system for learning. This means that bad retrieval quality in the language areas will now not be interpreted as uncertainty in the auditory input, but as evidence that a previously unknown object is going to be learnt. If the latter is the case, new representations for the object will be generated in the corresponding language areas. New assemblies will be generated from the strongest activated neurons, while a certain percentage of randomly chosen neurons may also be added to add some variability to the patterns. In parallel, the system uses its object classification to see whether the visual input is already known and if not, it extends its object recognition system by a new class for "cup". Finally, it associates the output of the object recognition system with the representations in the language areas to bind the different modalities for the new object.

After successful learning, the new object can be used like any of the previously stored objects, e.g. the robot can correctly respond to commands like "bot show cup".

6 Results

We have implemented the system described above on a robot. Running only the language system on a standard laptop machine (P4 1.5 GHz), it is able to process sentences much faster than one can actually speak or type the commands. Currently, the language model consists of 18000 neurons divided into 18 populations. The associative memories have a vocabulary of about 50 words. In our current implementation

we do not have a speech recogniser, so input must be typed in with the keyboard or alternatively one can directly activate some neurons in the input populations.

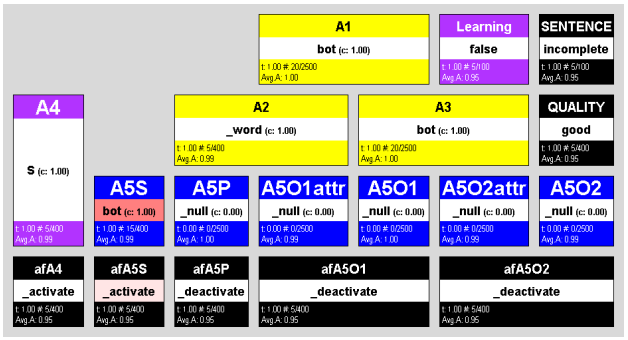

Figure 2: The language system after processing the input word "bot" as start of a new sentence.

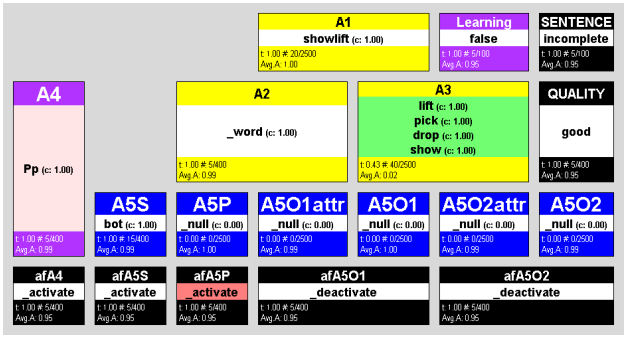

Figure 3: The language system after almost processing "bot show/lift", where "show/lift" is an artificial ambiguity between the words show and lift.

In the following we will show how the system deals with the sentence "bot show/lift green wall" with an artificial ambiguity between show and lift (parts of both representations are equally strong activated in the input). Figure 2 shows the system after the first word "bot" has been processed. Area A1 is the input area where the assembly representing the word "bot" is active. For convenience, the name of the active representation is displayed instead of neural activity. The activation is then forwarded to area A2/A3 which separate between simple grammatical structure elements (syntax) and their meaning (semantic). Thus, A2 has activated the "word" assembly while A3 represents the meaning "bot" again. As this is the first word of a sentence and currently all sentences have to start with a subject, area A5-S became activated and binds the word to its grammatical meaning in the current sentence.

In figure 3 the ambiguous input "show/lift" is presented. Area A1 has a "showlift" representation stored (which is actually a mixture of the show and lift assemblies), this is just for convenience to see that

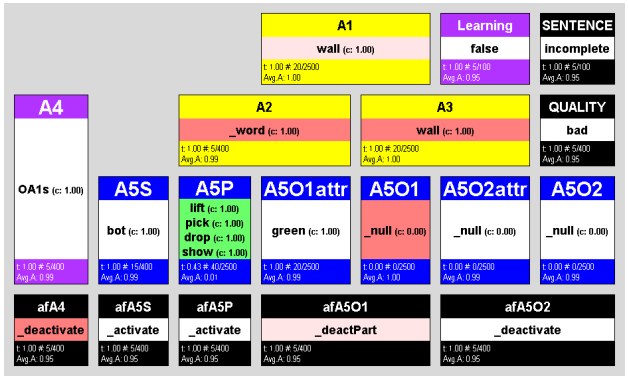

Figure 4: The language system after processing "bot show/lift green", where the final word "wall" is about to be activated in area A5-O1.

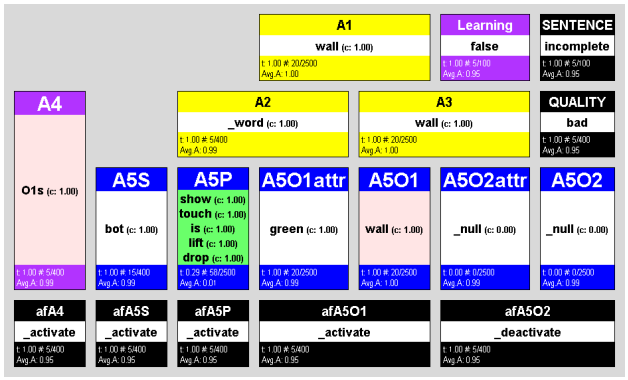

Figure 5: The language system while disambiguating "bot show/lift green wall". The disambiguation is already starting to be effective, the "show" representation moved up in A5P, which means that it spiked earlier than all the others.

really a mixture of both representations has been activated. All other populations only have representations for "show" and "lift" but do not know anything about a mixture of the both. Thus, as the activation gets forwarded to A3, the population notices ambiguity, lowers its α-parameter and activates multiple assemblies (here, "lift", "pick", "drop", and "show" are fully activated, as all four of them have large enough overlap with "lift" and "show"). As before, the activation is then processed and stored finally in area A5-P, as it is the predicate of the sentence.

In A5-P, the ambiguity is held while the input "green" is processed, figure 4 shows the system when already the last word "wall" comes in. In the next step, the "wall" assembly is activated in A5-O1, the object area of the language system. If this is the case, the disambiguation connection from A5-O1 to A5-P will give additional multiplicative input on area A5-P, where it privileges action verbs that can be done with large objects like a wall. Obviously, this is not the case for "lift" and therefore, "show" is going to win

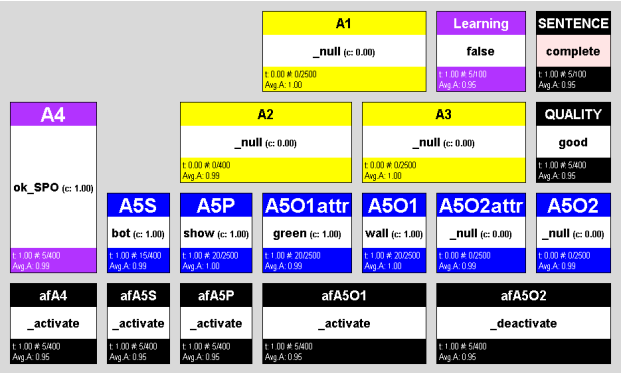

Figure 6: The language system after successfully interpreting the ambiguous input "bot show/lift green wall".

the battle. The disambiguating input is very weak, so it takes several time steps until area A5-P can do a final decision, figure 5 shows the system while the disambiguation takes place. In figure 6, the final state of the machine is depicted, showing that it correctly decided for "bot show green wall".

7 Conclusion

The neural implementation of the language understanding system presented here not only shows that this comparatively intricate logical task can be mastered by a neural network architecture in real time, it also gives some additional advantages in terms of robustness and context-awareness. Indeed the system can correct ambiguous input on the single word level due to the context of the whole sentence and even the complete sensory-motor situation. Similarly the language input could be used to disambiguate ambiguous results of visual object recognition.

This demonstrates the usefulness of a close interplay between symbolic and subsymbolic information processing (also known as "symbol grounding") in autonomous robots, which can be easily achieved by biologically inspired neural networks.

8 Acknowledgements

This work was partially supported by the MirrorBot project of the European Union, award #IST-2001-35282.

References

[1] F. Pulvermüller. Words in the brain's language. *Behavioral and Brain Sciences*, 22:253–336, 1999.

[2] D.O. Hebb. *The organization of behavior. A neuropsychological theory.* Wiley, New York, 1949.

[3] G. Palm. *Neural Assemblies. An Alternative Approach to Artificial Intelligence.* Springer, Berlin, 1982.

[4] G. Palm. Cell assemblies as a guideline for brain research. *Concepts in Neuroscience*, 1:133–148, 1990.

[5] G. Palm. On the internal structure of cell assemblies. In A. Aertsen, editor, *Brain Theory.* Elsevier, Amsterdam, 1993.

[6] G. Palm. Local rules for synaptic modification in neural networks. *Journal of Computational Neuroscience*, 1990.

[7] D.J. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–962, 1969.

[8] G. Palm. On associative memories. *Biological Cybernetics*, 36:19–31, 1980.

[9] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science, USA*, 79:2554–2558, 1982.

[10] G. Palm. Local learning rules and sparse coding in neural networks. In R. Eckmiller, editor, *Advanced Neural Computers.* North-Holland, Amsterdam, 1990.

[11] G. Palm. Memory capacities of local rules for synaptic modification. A comparative review. *Concepts in Neuroscience*, 2:97–128, 1991.

[12] G. Palm. On the information storage capacity of local learning. *Neural Computation*, 4:703–711, 1992.

[13] G. Palm and F.T. Sommer. Information capacity in recurrent McCulloch-Pitts networks with sparsely coded memory states. *Network*, 3:177–186, 1992.

[14] A. Knoblauch and G. Palm. Pattern separation and synchronization in spiking associative memories and visual areas. *Neural Networks*, 14:763–780, 2001.

[15] R. Fay, U. Kaufmann, A. Knoblauch, H. Markert, and G. Palm. Combining visual attention, object recognition and associative information processing in a neurobotic system. In Wermter et al. [17].

[16] H. Markert, A. Knoblauch, and G. Palm. Detecting sequences and understanding language with neural associative memories and cell assemblies. In Wermter et al. [17], pages 106–116.

[17] S. Wermter, G. Palm, and M. Elshaw, editors. *Biomimetic Neural Learning for Intelligent Robots.* Springer, Heidelberg, New York, 2005.