Detecting sequences and understanding language with neural associative memories and cell assemblies.

Heiner Markert, Andreas Knoblauch, Günther Palm

2005

Preprint:

This is an accepted article published in Biomimetic Neural Learning for Intelligent Robots. The final authenticated version is available online at: https://doi.org/[DOI not available]

Detecting Sequences and Understanding Language with Neural Associative Memories and Cell Assemblies

Heiner Markert[1], Andreas Knoblauch[1,2], 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, knoblauch, 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. Using associative memories and sparse distributed representations we have developed a system that can learn to associate words with objects, properties like colors, and actions. This system is used in a robotics context to enable a robot to respond to spoken commands like "bot show plum" or "bot put apple to yellow cup". This involves parsing and understanding of simple sentences and "symbol grounding", for example, relating the nouns to concrete objects sensed by the camera and recognized by a neural network from the visual input.

1 Introduction

When words referring to actions or visual scenes are presented to humans, distributed networks including areas of the motor and visual systems of the cortex become active (e.g. [1]). The brain correlates of words and their referent actions and objects appear to be strongly coupled neuronal assemblies in defined cortical areas. The theory of cell assemblies [2, 3, 4, 5, 6] provides one of the most promising frameworks for modeling and understanding the brain in terms of distributed neuronal activity. It is suggested that entities of the outside world (and also internal states) are coded in groups of neurons rather than in single ("grandmother") cells, and that a neuronal cell assembly is generated by Hebbian coincidence or correlation learning [7, 8] where the synaptic connections are strengthened between co-activated neurons. Thus models of neural (auto-)associative memory have been developed as abstract models for cell assemblies [9].

One of our long-term goals is to build a multi-modal internal representation for sentences and actions using cortical neuron maps, which will serve as a basis for the emergence of action semantics and mirror neurons [10, 11, 12]. We have

S. Wermter et al. (Eds.): Biomimetic Neural Learning, LNAI 3575, pp. 106–116, 2005.

developed a model of several visual, language, planning, and motor areas to enable a robot to understand and react to spoken commands in basic scenarios of the MirrorBot project [11, 12, 13, 14] that is described in the first part of this book. The essential idea is that different cortical areas represent different aspects (and correspondingly different notions of similarity) of the same entity (e.g., visual, auditory language, semantical, syntactical, grasping related aspects of an apple) and that the (mostly bidirectional) long-range cortico-cortical projections represent hetero-associative memories that translate between these aspects or representations. This involves anchoring symbols such as words in sensory and motor representations where invariant association processes are required, for example recognizing a visually perceived object independent of its position, color, or view direction. Since word symbols usually occur in the context of other words specifying their precise meaning in terms of action goals and sensory information, anchoring words is essentially equivalent to language understanding.

In this work we present a neurobiologically motivated model of language processing based on cell assemblies [2, 3, 4, 5]. We have developed a system that can learn to associate words with objects, properties like colors, and actions. This system is used in a robotics context to enable a robot to respond to spoken commands like "bot show plum" or "bot put apple to yellow cup". The scenario for this is a robot close to one or two tables on which there are certain kinds of fruit and/or other simple objects. We can demonstrate part of this scenario where the task is to find certain fruits in a complex visual scene according to spoken or typed commands. This involves parsing and understanding of simple sentences and relating the nouns to concrete objects sensed by the camera and recognized by a neural network from the visual input.

In the first section we outline the concept of cell assemblies as a model for sequential associative processing in cortical areas and how our model is related to discrete finite automates and sequence detector networks by Pulvermüller [1, 15], also explained in this book [16]. Then we briefly describe our robot architecture used for implementing simple scenarios of associating words to objects, and detail the language module. Finally, we summarize and discuss our results.

2 Language and Cell Assemblies

A large part of our model is based on associative memory and cell assemblies. Anchoring a symbol first requires understanding the context in which the symbol occurs. Thus, one requirement for our system is language processing and understanding.

2.1 Regular Grammars, Finite Automates, and Neural Assemblies

Noam Chomsky developed a hierarchy for grammar types [17, 18]. For example, a grammar is called *regular* if the grammar can be expressed by rules of the type

$$A \rightarrow a$$
$$B \rightarrow bC$$

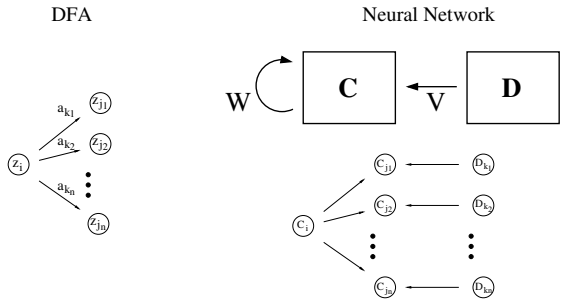

Fig. 1. Comparison of a deterministic finite automaton (DFA, left side) with a neural network (right side) implementing formal language. Each δ transition $\delta(z_i, a_k) = z_j$ corresponds to synaptic connections from neuron C_i to C_j and from input neuron D_k to C_j (see text for details)

where lower case letters are *terminal symbols* (i.e. elements of an alphabet Σ), and upper case letters are *variables*. Usually there is a starting variable S which can be expanded by applying the rules. A sentence $s \in \Sigma^*$ (which is a string of alphabet symbols of arbitrary length) is called *valid with respect to the grammar* if s can be derived from S by applying grammatical rules and resolving all variables by terminal symbols.

There are further grammar types in the Chomsky hierarchy which correspond to more complex rules, e.g. context-free and context-sensitive grammars, but here we will focus on regular grammars. It is easy to show that regular grammars are equivalent to deterministic finite automata (DFA). A DFA can be specified by $M = (Z, \Sigma, \delta, z_0, E)$ where $Z = \{z_0, z_1, ..., z_n\}$ is the set of states, Σ is the alphabet, $z_0 \in Z$ is the starting state, $E \subseteq Z$ contains the terminal states, and the function $\delta : (Z, \Sigma) \to Z$ defines the (deterministic) state transitions. A sentence $s = s_1 s_2 ... s_n \in \Sigma^*$ is valid with respect to the grammar if iterated application of δ on z_0 and the letters of s transfers the automaton's starting state to one of the terminal states, i.e., if $\delta(...\delta(\delta(z_0, s_1), s_2), ..., s_n) \in E$ (cf. left side of Fig. 1).

In the following we show that DFAs are equivalent to binary recurrent neural networks such as the model architecture described below (see Fig. 3). As an example, we first specify a simpler model of recurrent binary neurons by $N = (C, I, W, V, c_0)$, where $C = \{C_0, C_1, ..., C_n\}$ contains the local cells of the network, $D = \{D_1, D_2, ..., D_m\}$ is the set of external input cells, $W = (w_{ij})^{n \times n}$ is a binary matrix where $w_{ij} \in \{0, 1\}$ specifies the strength of the local synaptic connection from neuron C_i to C_j, and, similarly, $V = (v_{ij})^{m \times n}$ specifies the synaptic connections from input cell D_i to cell C_j. The temporal evolution of the network can be described by

$$c_i(t+1) = \begin{cases} 1, \text{if } \sum_j w_{ji} c_j(t) + \sum_j v_{ji} d_j(t) \geq \Theta_i \\ 0, \text{otherwise.} \end{cases}$$

where $c_i(t)$ is the output state of neuron C_i at time t, and Θ_i is the threshold of cell C_i. Figure 1 illustrates the architecture of this simple network.

The network architecture can easily be adapted to simulate a DFA. We identify the alphabet Σ with the input neurons, and the states Z with the local cells, i.e. each $a_i \in \Sigma$ corresponds to input cell D_i, and, similarly, each $z_i \in Z$ corresponds to a local cell C_i. Then we can specify the connectivity as follows: Synapses w_{ij} and v_{kj} are active if and only if $\delta(z_i, a_k) = z_j$ for the transition function δ of the DFA (see Figure 1). In order to decide if a sentence $s = a_{i(0)}a_{i(1)}a_{i(2)}...$ is valid with respect to the language we can specify the activation of the input units by $d_i(t) = 1$ and $d_j = 0$ for $j \neq i(t)$. By choosing threshold $\Theta_i = 2$ and a starting activation where only cell c_0 is active, the network obviously simulates the DFA. That means, after processing of the last sentence symbol, one of the neurons corresponding to the end states of the DFA will be active if and only if s is valid.

To be precise, the above algorithm represents a slight oversimplification. In this implementation it might happen that the next state is not determined uniquely. To avoid this problem it would be sufficient to use states representing pairs (z_i, a_k) of a state z_i and an action a_k leading to it. Then the synapses from (z_i, a_l) to (z_j, a_k) and from a_k to (z_j, a_k) are activated for the transition $\delta(z_i, a_k) = z_j$.

The described neural network architecture for recognizing formal languages is quite simple and reflects perfectly the structure of a DFA even on the level of single neurons. In addition, this network is also very similar to the network of sequence detectors discussed in [15, 16]. Essentially an elementary sequence detector can be identified with a 3-state automaton, where the sequence ab is identified by two transitions from an initial state z_0 into a final state z_2 as depicted in the left part of figure 2. The resulting network (depicted in the right part of figure 2) resembles the sequence detector idea of Pulvermüller ([1, 15]) and they obey the same underlying principle: Pulvermüller's simplest linear sequence detector (e.g. for ab) needs a weak input for a followed by a strong input for b. This works because the strong input decays faster then the weak input (in absolute terms; in relative terms they both decay exponentially fast). Thus Pulvermüller's idea works whenever the second input b decays faster than the first input a. This makes the sequence ab more effective than ba, because more activity is left from a when b occurs, than vice versa. This simple principle also holds for an automaton (see figure 2) and the corresponding network, because a feeds back onto itself (it is an auto-associative assembly) and therefore is more

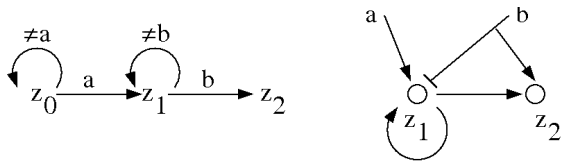

Fig. 2. Identification of an elementary sequence detector representing the sequence ab with a 3-state automaton. Left: schematic diagram, right: neural implementation, where $\rightarrow$ means excitation and $\dashv$ means inhibition

persistent than b. Ideally, this scheme works for arbitrary delays between a and b, in contrast to Pulvermüller's simple sequence detector that only works for a small range of delays ([16]). The idea of using auto-associative persistent patterns with different persistency in different areas is used extensively in the model derived in the next section. Also the sequencing or automaton idea is used at least in one area (A4, see figure 3), where both auto-associative memories stabilizing individual patterns and hetero-associative memories representing patterns sequences are stored in the same local network.

Coming back to the cell assembly perspective, clearly a network as in figure 1 or 2, where single neurons are used to code the different states, is biologically not very realistic, since, for example, such an architecture is not robust against partial destruction and it is not clear how such a delicate architecture could be learned. The model becomes more realistic if we interpret the nodes in figure 1 or figure 2 not as *single* neurons but as groups of nearby neurons which are strongly interconnected, i.e., as local cell assemblies. This architecture has two additional advantages: First, it enables *fault tolerance* since incomplete input can be completed to the whole assembly. Second, overlaps between different assemblies can be used to express similarity, hierarchical and other relations between represented entities. In the following subsection we briefly describe a model of associative memory which allows to implement the assembly network analogously to the network of single neurons in figure 1 or 2.

2.2 Cell Assemblies and Neural Associative Memory

We decided to use the *Willshaw associative memory* [19, 20, 4, 21, 22, 23] as a single framework for the implementation of cell assemblies in cortical areas. A *cortical area* consists of n binary neurons which are connected with each other 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 in the synaptic connectivity such that any two neurons of an assembly are bidirectionally connected. Thus, an assembly consisting of k neurons can be interpreted as a k-clique in the graph corresponding to the binary matrix A of synaptic connections. This model class has several advantages over alternative models of associative memory such as the most popular Hopfield model [24]. For example, it better reflects the cortical reality where it is well known that activation is sparse (most neurons are silent most of the time), and that any neuron can have only one type of synaptic connection (either excitatory or inhibitory).

Instead of classical one-step retrieval we used an improved architecture based on spiking associative memory [25, 13]. A cortical area is modeled as a local population of n neurons which receive input from other areas via Hebbian learned hetero-associative connections. In each time step this external input initiates pattern retrieval. The neurons receiving the strongest external input will fire first, and all emitted spikes are fed back immediately through the Hebbian learned auto-associative connections resulting in activation of single assemblies. 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.

In the following section we present the architecture of our cortical model which enables a robot to associate words to visually recognized objects, and thereby anchoring symbolic word information in sensory data. This model consists of a large number of interconnected cortical areas, each of them implemented by the described spike counter architecture.

3 Cell-Assembly Based Model of Cortical Areas

We have designed a cortical model consisting of visual, tactile, auditory, language, goal, and motor areas (see figure 3), and implemented parts of the model on a robot. Each cortical area is based on the spike counter architecture described in the previous section. The model is simulated synchronously in discrete time steps. That means, in each time step t each area computes its output vector $y(t)$ as a function of the output vectors of connected areas at time $t-1$. In addition to the auto-associative internal connection within each area there are also hetero-associative connections between theses areas (see figure 4).

3.1 Overall Architecture

Figure 3 illustrates the overall architecture of our cortical model. The model consists of auditory areas to represent spoken or typed language, of grammar areas to interpret spoken or typed sentences, visual areas to process visual input, goal areas to represent action schemes, and motor areas to represent motor output. Additionally, we have auxiliary areas or fields to activate and deactivate the cortical areas (activation fields), to compare corresponding representations in different areas (evaluation fields), and to implement visual attention.

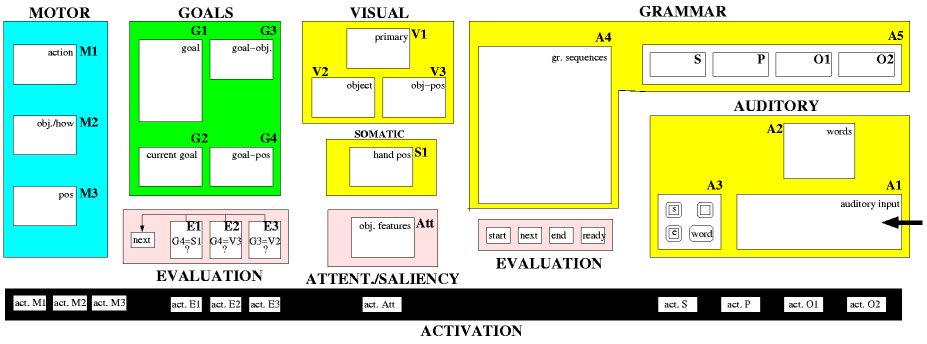

Fig. 3. Cortical architecture involving several inter-connected cortical areas corresponding to auditory, grammar, visual, goal, and motor processing. Additionally the model comprises evaluation fields and activation fields (see text)

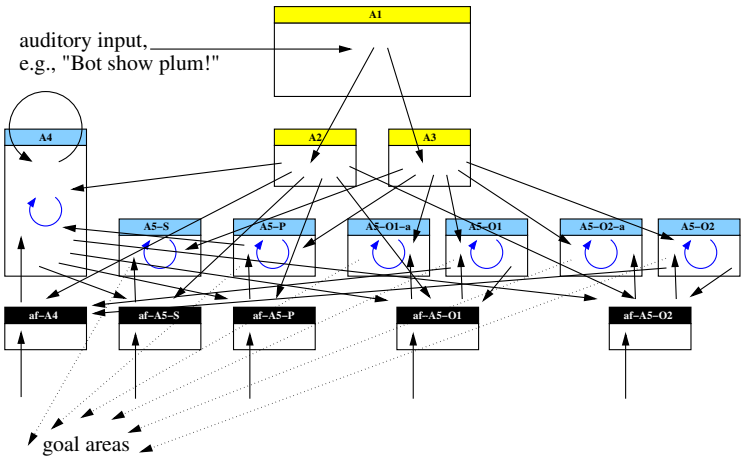

Fig. 4. The language system consisting of 10 cortical areas (large boxes) and 5 thalamic activation fields (small black boxes). Black arrows correspond to inter-areal connections, gray arrows within areas correspond to short-term memory

Each small white box corresponds to an associative memory as described in the previous section. The auditory areas comprise additional neural networks for processing of acoustic input, i.e. they perform basic speech recognition. The main purpose of the visual fields is to perform object recognition on the camera image.

Currently, we have implemented most parts of the model on a robot. The object recognition system basically consists of three components:

1. The *visual attention control system* localizes the objects of interest based on an attention control algorithm using top-down information from higher cortical areas.
2. The *feature extraction system* analyzes a window taken from the camera image corresponding to the region of interest. Scale and translation invariance is achieved by rescaling the window and using inherently invariant features as input for the classification system. The extracted features comprise local orientation and color information.
3. The *classification system* uses the extracted features as input to a hierarchical neural network which solves the classification task. The basic idea of using hierarchical neural networks is the division of a complex classification task into several less complex classification tasks by making coarse discrimination at higher levels of the hierarchy and refining the discrimination with decreasing depth of the hierarchy. Beneficial side effects of the hierarchical structure are the possibility to add additional classes quite easily at run-time, which means that the system will be able to learn previously untrained objects online, and the possibility to investigate the classification task at intermediate states. The latter can be useful if, for example, the full

classification information is not needed for the task at hand, or simply to gain more insight in the performance of the network.

For a more detailed description of the whole model, see [11] in this book. For additional detailed information on the object recognition system see for example [26].

3.2 Language Processing

Figure 4 gives an overview of our model for cortical language processing. It basically implements a sequence detector network or DFA for a previously defined regular grammar (as in figure 1). The system presented here can understand simple sentences like "bot show red plum" or "bot lift apple".

Areas A1, A2 and A3 are primary auditory areas, A1 represents auditory input by primary linguistic features, whereas area A2 and A3 classify the input with respect to function and content, respectively. Areas A2 and A3 serve as input areas to the sequence detector circuit.

Essentially, areas A4 and A5-X implement the DFA. The sequence detector network is split into several areas to enable the model to keep track of the path of states the automaton took to achieve the final state. This leads to a very useful representation of the parsed sentence in the end where each of the areas A5-X is filled with one word in one special grammatical context (e.g. A5S holds the subject of the sentence). Further interpretation of the semantics of the sentence then becomes relatively easy ([11] shows how our model for action planning uses this information).

Areas A5-X stores the state (z_i) the automaton has reached, together with the input (a_k) leading to that state. In our example, the inputs a_k are words and the states basically reflect the grammatical role of the input word in the sentence. The A5-X areas are used to explicitly store the input words a_k that lead to their activation. They implicitly store the grammatical context, because subjects will be stored in area A5S, predicates in A5P and so on. This corresponds to storing the pairs (z_i, a_k) for each state of the automaton.

The possible state transitions are stored in area A4. Hetero-associative feedback connections with longer delay from area A4 onto itself represent the possible state transitions, the current state is kept persistent by auto-associative feedback with short delays. To perform a state transition, the whole area A4 is inhibited for a short while, eliminating the effect of the auto-associative feedback. Because of the longer delay, the hetero-associative connection will still be effective if the inhibition is released. Biased by the new input pattern, it will then switch to the next state according to the next input symbol.

Area A4 only represents the state transition matrix; the usage of pairs in A4 is not necessary in this model because the required information is represented in areas A5-X, which all project onto area A4 making the state transitions unique.

Figure 5 shows the state of the system after processing the input sequence "bot put plum (to) green apple", where "to" does not belong to the actual input sequence due to simplification of the grammatical rules. The input sentence is

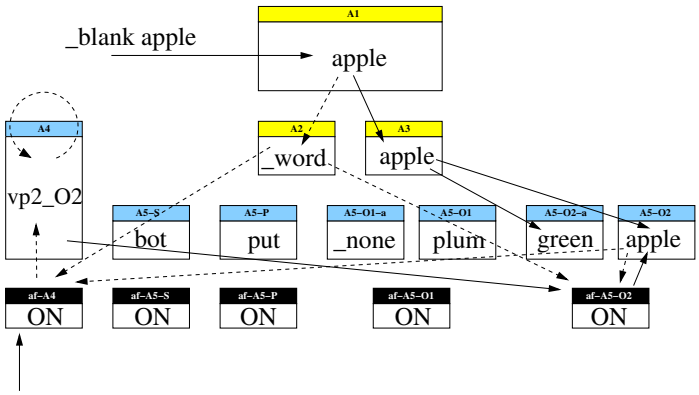

Fig. 5. System state of the language model after 30 simulation steps when processing the sentence "Bot put plum to green apple". (Processing of a word requires about 5-6 steps on average; during each simulation step the state of the associative network is synchronously updated)

segmented into subject, predicate and the two objects, and the automaton will reach its final state when the input "apple" disappears and a symbol for the end of the input sentence will be activated in area A1 and A2.

4 Conclusion

We have presented a cell assembly based model for cortical language processing that can be used for associating words with objects, properties like colors, and actions. This system is used in a robotics context to enable a robot to respond to spoken commands like "bot put plum to green apple". The model shows how sensory data from different modalities (e.g., vision and speech) can be integrated to allow performance of adequate actions. This also illustrates how symbol grounding could be implemented in the brain involving association of symbolic representations to invariant object representations.

Although we have currently stored only a limited number of objects and sentence types, it is well known for our model of associative memory that the number of storable items scales with $(n/\log n)^2$ for n neurons [19, 20, 21]. However, this is true only if the representations are sparse and distributed which is a design principle of our model. As any finite system, our language model can implement only regular languages, whereas human languages seem to involve context-sensitive grammars. On the other hand, also humans cannot "recognize" formally correct sentences beyond a certain level of complexity.

The neural implementation of this language understanding system 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, we have shown at the Neu-

roBotics Workshop [11, 12] that this 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.

For example the sentence "bot lift bwall" with an ambiguous word between "ball" and "wall" is correctly interpreted as "bot lift ball", because a wall is not a liftable object. Similarly, a sentence like "bot show/lift green wall" with an artificial ambiguity between "show" and "lift", can be understood as "bot show green wall", even if the disambiguating word "wall" comes later and even across an intermittent word ("green"). Similarly the language input could be used to disambiguate ambiguous results of visual object recognition, and vice versa.

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.

References

1. Pulvermüller, F.: Words in the brain's language. Behavioral and Brain Sciences **22** (1999) 253–336
2. Hebb, D.: The organization of behavior. A neuropsychological theory. Wiley, New York (1949)
3. Braitenberg, V.: Cell assemblies in the cerebral cortex. In Heim, R., Palm, G., eds.: Lecture notes in biomathematics (21). Theoretical approaches to complex systems. Springer-Verlag, Berlin Heidelberg New York (1978) 171–188
4. Palm, G.: Neural Assemblies. An Alternative Approach to Artificial Intelligence. Springer, Berlin (1982)
5. Palm, G.: Cell assemblies as a guideline for brain research. Concepts in Neuroscience **1** (1990) 133–148
6. Palm, G.: On the internal structure of cell assemblies. In Aertsen, A., ed.: Brain Theory. Elsevier, Amsterdam (1993)
7. Palm, G.: Local rules for synaptic modification in neural networks. Journal of Computational Neuroscience (1990)
8. Palm, G.: Rules for synaptic changes and their relevance for the storage of information in the brain. Cybernetics and Systems Research (1982)
9. Palm, G.: Associative memory and threshold control in neural networks. In Casti, J., Karlqvist, A., eds.: Real Brains - Artificial Minds. North-Holland, New York, Amsterdam, London (1987)
10. Rizzolatti, G., Fadiga, L., Fogassi, L., Gallese, V.: Resonance behaviors and mirror neurons. Archives Italiennes de Biologie **137** (1999) 85–100
11. Fay, R., Kaufmann, U., Knoblauch, A., Markert, H., Palm, G.: Combining visual attention, object recognition and associative information processing in a neurobotic system. [27]
12. Fay, R., Kaufmann, U., Knoblauch, A., Markert, H., Palm, G.: Integrating object recognition, visual attention, language and action processing on a robot in a neurobiologically plausible associative architecture. accepted at Ulm NeuroRobotics workshop (2004)
13. Knoblauch, A.: Synchronization and pattern separation in spiking associative memory and visual cortical areas. PhD thesis, Department of Neural Information Processing, University of Ulm, Germany (2003)

14. Knoblauch, A., Fay, R., Kaufmann, U., Markert, H., Palm, G.: Associating words to visually recognized objects. In Coradeschi, S., Saffiotti, A., eds.: Anchoring symbols to sensor data. Papers from the AAAI Workshop. Technical Report WS-04-03. AAAI Press, Menlo Park, California (2004) 10–16
15. Pulvermüller, F.: The neuroscience of language: on brain circuits of words and serial order. Cambridge University Press, Cambridge, UK (2003)
16. Knoblauch, A., Pulvermüller, F.: Sequence detector networks and associative learning of grammatical categories. [27]
17. Hopcroft, J., Ullman, J.: Formal languages and their relation to automata. Addison-Wesley (1969)
18. Chomsky, N.: Syntactic structures. Mouton, The Hague (1957)
19. Willshaw, D., Buneman, O., Longuet-Higgins, H.: Non-holographic associative memory. Nature **222** (1969) 960–962
20. Palm, G.: On associative memories. Biological Cybernetics **36** (1980) 19–31
21. Palm, G.: Memory capacities of local rules for synaptic modification. A comparative review. Concepts in Neuroscience **2** (1991) 97–128
22. Schwenker, F., Sommer, F., Palm, G.: Iterative retrieval of sparsely coded associative memory patterns. Neural Networks **9** (1996) 445–455
23. Sommer, F., Palm, G.: Improved bidirectional retrieval of sparse patterns stored by hebbian learning. Neural Networks **12** (1999) 281–297
24. Hopfield, J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Science, USA **79** (1982) 2554–2558
25. Knoblauch, A., Palm, G.: Pattern separation and synchronization in spiking associative memories and visual areas. Neural Networks **14** (2001) 763–780
26. Fay, R., Kaufmann, U., Schwenker, F., Palm, G.: Learning object recognition in an neurobotic system. acctepted at 3rd workshop SOAVE2004 - SelfOrganization of AdaptiVE behavior, Illmenau, Germany (2004)
27. Wermter, S., Palm, G., Elshaw, M., eds.: Biomimetic Neural Learning for Intelligent Robots. Springer, Heidelberg, New York (2005)