MVA'94 IAPR Workshop on Machine Vision Applications Dec. 13-15, 1994, Kawasaki

Generation of a Position Tolerant Representation of Edges

B. Mertsching[1] and J. Schnusenberg
University of Hamburg
Department of Computer Science/AG IMA
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany

S. Neusser and T. Schwederski
Institute for Microelectronics Stuttgart
Allmandring 30A, 70569 Stuttgart, Germany

Abstract

Within the SENROB-project[2] (sensor driven robot-vision system), a robot-supported image analysis system has been developed that is able to recognize arbitrarily oriented and positioned workpieces with the help of a camera mounted on a gripping device. To achieve a high recognition rate even in case of lateral displacements, e.g. fovealisation failures, a position tolerant representation of edges, called edge clouds, is used. This approach is motivated by similar processes in the visual system of animals. The real-time use in a robot system and the complexity of the algorithms require the implementation of a massively parallel VLSI system.

The entire task can be subdivided into two parts: In a first step, the system extracts oriented edge elements from the preprocessed image, which will be used to calculate the edge clouds. The second step, the generation of edge clouds, consists of the detection of continuous edges in a window and the generalization of these sequences with respect to their orientation. Two methods for generating edge clouds are conceived and proper VLSI architectures for their implementation[3]. A hierarchical and a border oriented approach are presented in this paper.

1 Motivation

The generation of a position tolerant representation of edges is motivated by the increase of robustness in the subsequent recognition process. In this paper, we describe a very effective representation of contours in gray scale images and their hardware realization. Our work aims at a robust, real-time robot-vision system (SENROB) in which we have incorporated neurobiologically influenced methods to achieve orientation and distance invariant object recognition.

In our system, color images are grabbed via a camera mounted at the hand of a robot. We use special focussing techniques to localize objects in the scene and sample them subsequently by an artificial retina which is a special form of logarithmic retinae.

Thereby, a selection of ring shaped representations from a superposition of differently scaled logarithmic retinae results into four spatial frequency channels. In these images, several features as e.g. edges, coloured regions, and higher characteristics are extracted and serve - combined with orientation information derived from the retinal structure and easily available distance information -as input for mapping processes producing normalized object representations which are presented to an associative network. By these means, once learnt objects can be recognized, while unknown ones are learnt. A detailed description of the system is out of the scope of this paper; we refer to [Hartmann 1994, Mertsching 1994]. Here, we concentrate on the representation of edge features and their generation.

Our space-variant system uses foveal sensing which is advantageous due to a wide angle field with a high resolution in its center and decreasing resolution towards its periphery. Unfortunately, slight focussing errors cannot be excluded. The feature 'coloured region' is resonably insensitive to this kind of error (and thus not regarded in this paper), but not the contour features. This leads to a position tolerant representation of edges and corners.

2 Contour Representation

The gray scale spatial frequency channels are mapped to a layer of artificial on-center (C_k^+) and off-center (C_k^-) neurons. They are created by center-surround coupling and their activity patterns are evaluated by a layer of simplified edge detector neurons.

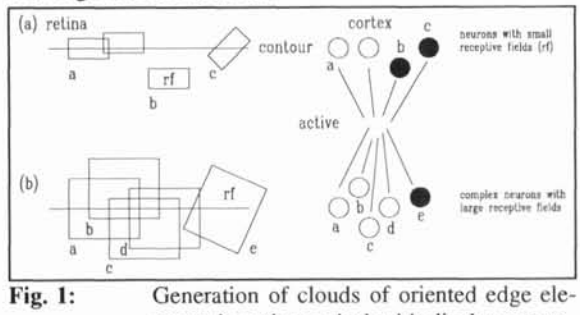

Fig. 1: Generation of clouds of oriented edge elements in order to deal with displacements

Their representation, similiar to the cortical representation

---

1 mertsching@informatik.uni-hamburg.de

2 Supported by the Federal German Ministry for Research and Technology (BMFT), grant 413-5839-01 IN 105/C5, the project is led by Prof. Hartmann, University of Paderborn, Germany

3 Supported by the German Research Association (DFG), grant ME 1289/1 and SCH 575/1

done by simple neurons in the primary visual cortex of cats (compare [Hubbel 1959]) provides "strings of oriented elements" (Fig. 1a). Previous work on the VLSI-implementation of the edge detector level was published in [Bilau 1993] and was slightly modified later. The generated strings are extremely sensitive to lateral displacements. For this reason, we generate a representation similiar to that by complex cortical neurons (Fig. 1b) which have distinctly enlarged receptive fields. Sets of artificial complex neurons with differently oriented receptive fields are assigned to each pixel of the channels. Due to the large sized receptive fields, a piece of contour activates a "cloud" of correctly oriented neurons assigned to neighboring pixels. In spite of unavoidable displacements of the edge clouds between learning and recognition, we can obtain high similarity measures. More about this layer can be found in [Hartmann 1990 a/b/c, 1991]. The hardware implementation of the complex neurons has been investigated recently. While the Paderborn group has applied a hierarchical approach to the problem, the Stuttgart team has used a border oriented one. Both approaches are presented in chapter 3 and 4.

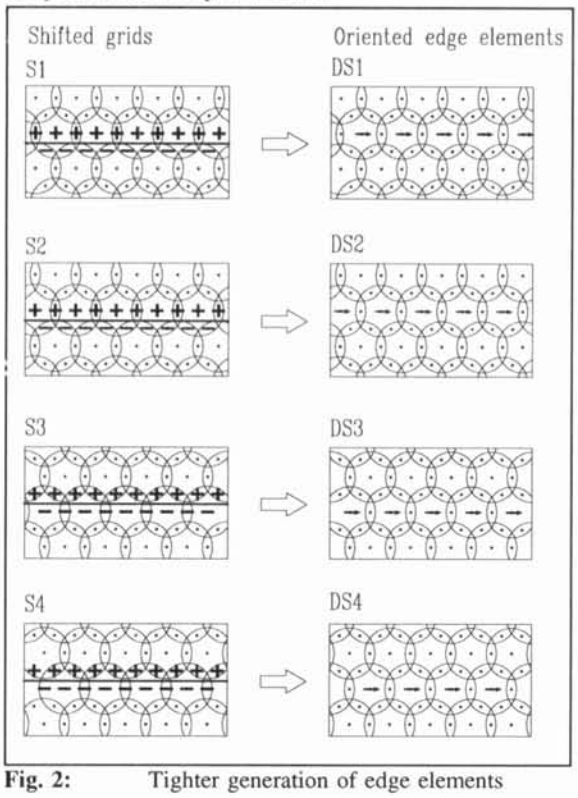

Fig. 2: Tighter generation of edge elements

3 Hierarchical Algorithm for Position Tolerant Edge Representation

The generation of position tolerant edge detectors is based on the generation of the oriented edge elements related to the simple cells. These are generated for islands of 7 pixels on a hexagonal grid and positioned at every second pixel of an image. To gain a position tolerant edge representation, a two step procedure is applied. In a first step, the density of edge elements has to be extended. As the detection uses every second pixel, the extension to every pixel leads to a tight representation of an edge sequence. A continuity check and generalization produce further redundancy, therefore wider edge clouds and results in higher robustness of the processing in the subsequent associative netzwork. The next two paragraphs describe the two step procedure using a horizontal edge sequence as an example.

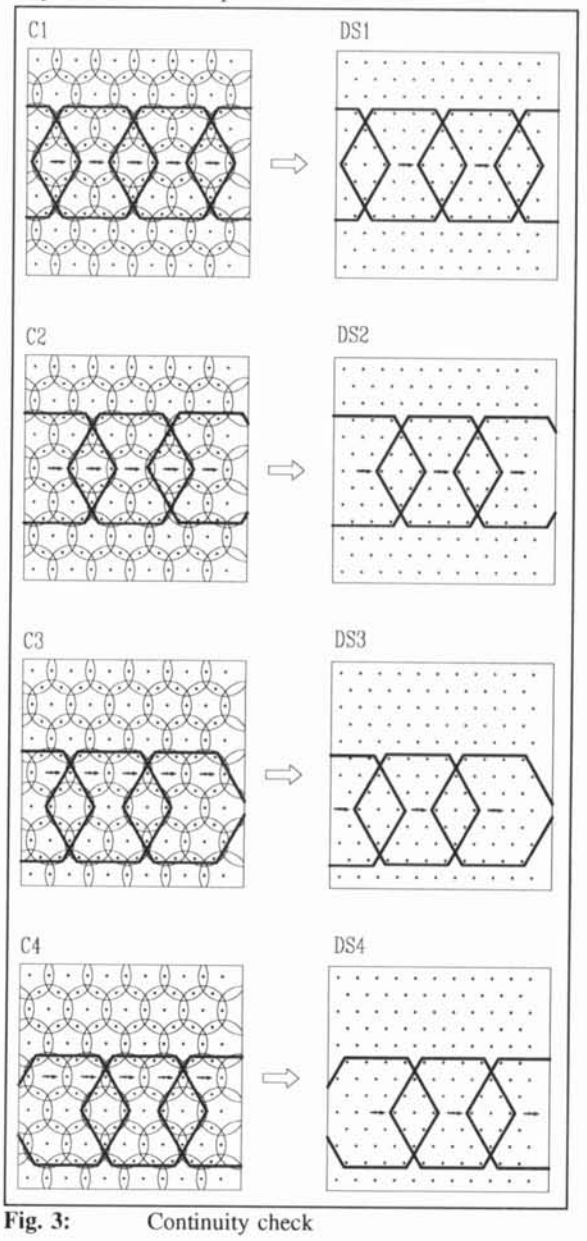

Fig. 3: Continuity check

3.1 Positon Tolerant Edge Representation by Tight Sampling

The first step of the generation of position tolerant edge representations is a tighter setting of the edge elements. Now every pixel of an image can possess such elements. The resulting strong overlap of the islands is motivated by the fact that the same structure is found by the receptive fields of the simple cortical cells. This generation of redundancy enlarges certainly the robustness of the recognition process, but the continuity check and the generaliza-

tion will improve the results furthermore. Fig. 2 explains the procedure in detail. Starting from the grid S1 as reference, the grids (S2-S4) are shifted by one pixel. One gains four data structures (DS1-DS4) with oriented edge detectors. Superimposing the results of the four data structures, the width of the simple edge sequence is doubled.

3.2 Position Tolerance by Continuity Check

The generalization of edge sequences by a continuity check has two purposes: on the one hand, short edge sequences that originate from interferences and that have no significance for the recognition process, are suppressed. On the other hand, the continuity check expands the edge clouds further. As we use a hexagonal grid, the continuity check works within a hexagonal window covering seven islands (Fig. 3). One condition for the continuity check is that a continuous edge sequence has to pass the window. If this condition is fulfilled a generalized edge element is assigned to the central pixel of the window.

According to the tight sampling, the window for the continuity check is placed on four different island positions. This procedure of continuity check is executed for the data structures DS1 to DS4. In this way, 16 separate data structures are built up. This algorithm can be used iteratively to get clouds of oriented edge elements with an arbitrary width.

4 The Border Oriented Algorithm for Position Tolerant Edge Representation

In the border oriented approach, detection and generalization of edge sequences in the vicinity of the central pixel will be performed in a single step.

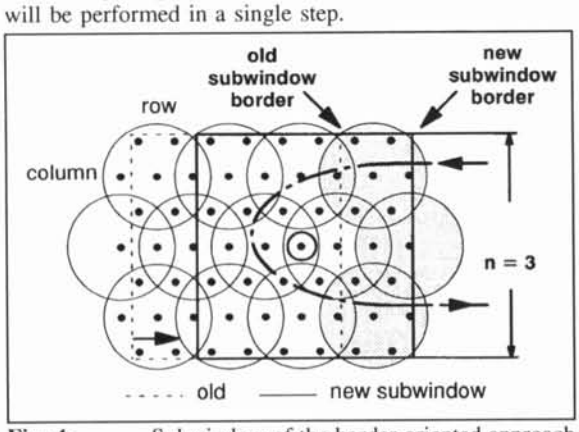

Fig. 4: Subwindow of the border oriented approach shifted by one edge element. The new additional row of edge element information is shaded.

The approach is based on to the fact that moving the observed subwindow by one border element in horizontal or vertical direction will add only a small amount of new information to the existing data. Results of former operations can be stored and used for the evaluation of the new subwindow (Fig. 4).

The approach reduces the evaluation complexity from $O(n^2)$ to $O(n)$ edge elements and minimizes hardware cost

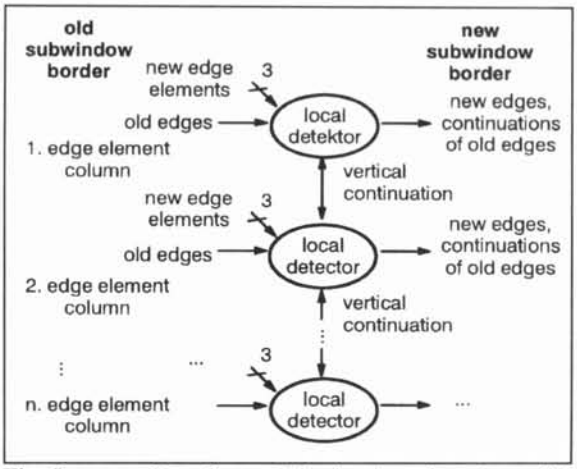

Fig. 5: Data flow of the border oriented parallel algorithm. Only local operations are used to search for edge continuations.

by using hardware the size of which increases linear with subwindow dimensions. The decrease in hardware area can be spent to increase the window size to the expected width of the edge cloud, so that processing of the image in a single run is enabled.

By applying this approach, the main task that has to be performed is searching for new edges or continuations of already detected edges in the added element row. Problem analysis shows that for most edge sequences, a local operating continuation detector is sufficient. Only vertical extensions over one or more columns result in a global information exchange. Even in this case, the information exchange can be split into local column to column operations. Fig. 5 depicts the resulting algorithm. Each continuation detection task works independent of its neighbors; it is fed by local available data only and can be performed in parallel. The results of each calculation cycle form the basis of the following evaluation.

5 Architecture of the Robot-Vision System

Fig. 6 presents an overview of the SENROB system. The whole process is roughly divided into the steps of preprocessing, the logarithmic-polar transformation, the generation of the laplacian image, the generation of oriented edge elements and edge clouds, the continuity check and a module for the data transformation and selection. The last module contains the invariances and the associative network.

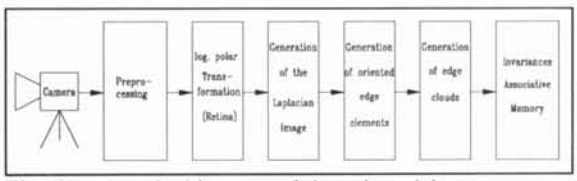

Fig. 6: Architecture of the robot-vision system.

The modules for building the sampling window and for the generation of edge detectors have been already realized and integrated in a TIP-System (**T**ransputer **I**mage **P**rocessing) from PARSYTEC. The hardware architec-

tures of these modules have been published in several papers [Bilau 1993, Bilau 1994]. Here, we focus on the modules for the generation of the edge clouds.

For the huge amount of image data, a dataflow architecture is well suited for this kind of application.
Constraints for the architecture are:

- the processing time has to be independent of the image data,
- the time delays from image grabbing to object recognition have to be as short as possible (delays of severals images are not acceptable).

6 Realization of the Hierarchical approach

As in chapter 3 described, the hierarchical approach uses at each level a processing window with a fixed size of seven islands. At first, the cross pointers for the three possible edge elements of the central island are generated. The cross pointer is the number of an edge element in any of the six neighboring islands, to which the central edge element is connected to. Second, the edge elements of the seven islands are combined and generalized to a new element. Discontinuous edge sequences are suppressed by this operation. Afterwards, the generalized edge elements represent a longer edge sequence and are interpreted as the data of an island for the next hierarchical level. Here, again a window of seven islands is formed, the data is combined and generalized to a new edge element.

6.1 Continuity Check Module

Fig. 7 illustrates the hierarchical architecture of the continuity check module without the controling units. The data from the module for generation of oriented edge elements is stored in a three rows wide FIFO buffer, the cross pointer module starts and the results are buffered. The subsequent module processes the generalized edge elements of the first hierarchy level. Again a row buffer is connected to the output of the module. To build up the edge elements at the next level, one has to connect four of the generalization modules to the output of the row buffer. The interweaved data structures of the first level must be fed to these modules in a demultiplexed manner to fit to the processing scheme of the hierarchical approach.

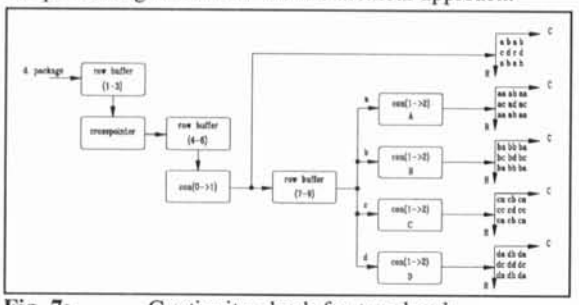

Fig. 7: Continuity check for two levels

6.2 Cross Pointer Module

The module (Fig. 8) generates the cross pointers for edge elements in block **A**. To generate the cross pointers of the edge elements in block **A**, the elements of the neighboring islands are stored in the blocks **B** up to **G** in separate pipeline registers.

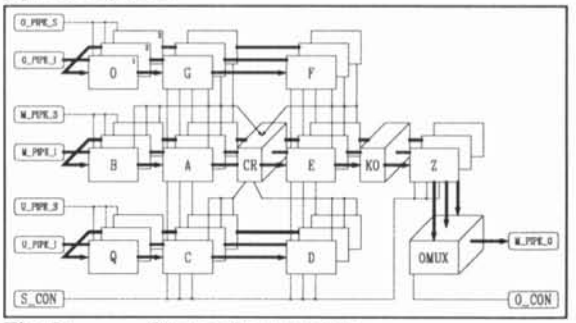

Fig. 8: Cross pointer module

A connection network generates the cross pointers, which are stored with the next pipeline clock in block **Z**. So at every pipeline clock, the processing window is moved by one island and the cross pointers are generated for the block **A**.

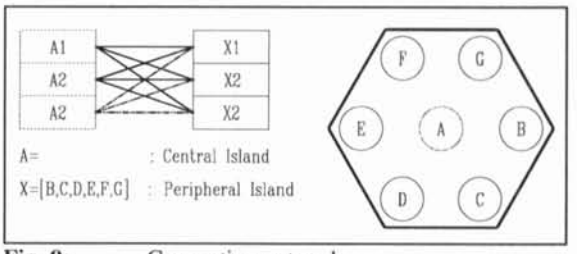

Fig. 9: Connection network

Every pipeline block is designed to store three edge elements due to a maximum of three edge elements at an island. To separate these, the connection network (Fig. 9) for the crosspointers is used. The results are stored at the next pipeline clock in the block **Z**.

6.3 Generalization Module

The cross pointer module as a preprocessing unit, builds up the information for the central edge element to which edge element in the neighborhood it is assigned to.
The next processing step is to combine these elements to a sequence that is generalized to an element at the next hierarchical level. This task is performed by the generalization module. This time consuming task is mapped onto two pipeline stages to allow an overlapping processing.

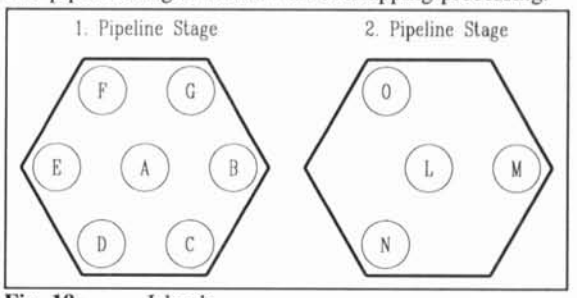

Fig. 10: Islands

The input data is built by the data of three rows. The first pipeline step eleminates the data of the islands **C**, **E** and **G** and assigns the data to any of the neighboring islands **A**, **B**, **D** and **F**, which possess additional comparators. To

achieve this, every edge element is applied sequentially to a bus, that is connected to the comparators of the islands **A**, **B**, **D** and **F**. If any of the comparators supplies a hit (a connection exists), the corresponding edge element is buffered into one of the island **A**, **B**, **D** and **F**. This pipeline step needs at least nine cycles to terminate. Afterwards; **A**, **B**, **D** and **F** contain edge sequences with up to two elements.

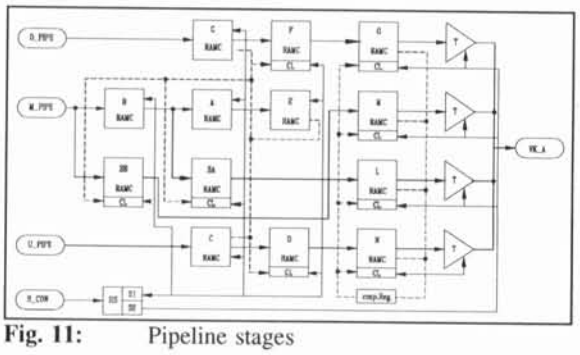

Fig. 11: Pipeline stages

The next pipeline clock buffers these sequences into the second pipeline stage for further processing. Every island, now labeled with **L**, **M**, **N** and **O**, obtains a comparator logic. An extra register is provided to enable the combination of two edge sequences, that are buffered in the same island. At least 12 cycles are required for this pipeline stage to terminate, because in the worst case 12 edge sequences have to be stored in these blocks. Now, the generalized edge element can be loaded into the subsequent row buffers.

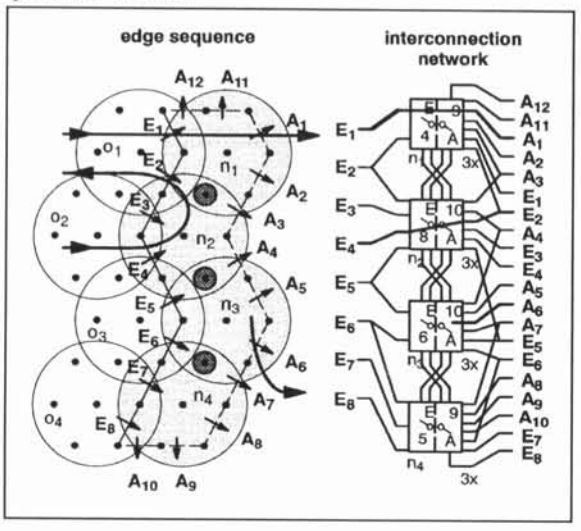

Fig. 12: Edge continuation situation at the window border and corresponding configuration of the interconnection network. E_X and A_X label possible entry and exit points of edges or inputs and outputs of the network, respectively. The shaded circled points indicate data bits in the edge elements that are used to configure the switches for vertical interconnections.

7 Realization of the Border Oriented Approach

The successful partition of the problem suggests a VLSI architecture as the second approach based on the consecutive transmission of edge data at the subwindow border through an dedicated unidirectional interconnection network. Hereby, an effective evaluation of edge sequences and a high degree of parallelism is obtained. As can be seen in Fig. 12, various hardware aspects are applied to implement the algorithm. Thus, the inputs and outputs of the interconnection network correspond to possible spatial entry and exit points of edges in the new edge element row.

The switching elements represent the column positions of edge elements. Their configuration is determined by the data of the edge elements and reflects the actual edge continuation. The interconnection net is duplicated to handle the left or right hand sided flow of edge sequences simultaneously. One transmitted data bit indicates this direction. Other transmitted and received data are used to identify each edge sequence uniquely. The ordered sequences are stored in a first-in first-out memory (FIFO) and the medium orientation is calculated incrementally.

8 Summary

In this paper we have presented two algorithms for the generation of a position tolerant representation of edges, which are currently investigated. The hierarchical approach can be cascaded to get a position tolerant representation of edges with an arbitrary width. The border oriented approach minimizes the hardware costs, which increases in a linear manner with the window dimension. Both approaches have to be examined in detail to decide, which one fulfills the constraints of processing time at a minimum of hardware costs.

9 References

[Bilau 1992]
Bilau, N.; Schnusenberg, J.; Ein schneller Codierungsprozessor für ein System zur echtzeitnahen Generierung des Hierarchischen Strukturcodes (HSC) mit Schnittstelle zum Erkennungssystem PANTER. In: Fuchs, S.; Hoffmann, R. (ed.): Mustererkennung 1992. Informatik aktuell. Berlin u. a. (Springer Verlag) 1992, pp. 310-315

[Bilau 1993]
Bilau, N; Schnusenberg, J.; Hartmann, G.; Siggelkow, A.; Schwederski, T.: A VLSI-Processor for the Generation of the Hierarchical Structure Code in Real-Time. In: Bayoumi, M. A.; Davis, L. S.; Valvanis, K. P. (ed.): CAMP '93: Computer Architecture for Machine Perception, New Orleans, 1993, pp. 67-76

[Mertsching 1994]
Mertsching, B.; Drüe, S.; Hartmann, G.: Robot Vision System-Learning and Recognizing Arbitrarily Located Objects from Different Camera Positions. In: Tanik, Murat M. (ed.): Computer Science and Technology, PD-Vol. 59. 1994, pp. 12-130

[Hartmann 1990a]
Hartmann, G.; Drüe, S.: Feature linking by synchronization in a two dimensional network. In: Caudill, M. (ed.): Theory Track: Neural & Cognitive Sciences of the Proceedings of the International Joint Conference on Neural Networks (IJCNN) 1990. Vol. 1, pp. 247-250

[Hartmann 1990b]
Hartmann, G.; Drüe, S.: Self Organization of a Network Linking Features by Synchronization. In: Eckmiller, R.; Hartmann, G. and G. Hauske (ed.): Parallel Processing in Neural Systems and Computer. Amsterdam u. a. (Elsvier Science Publisher/North-Holland) 1990, pp. 361-364

[Hartmann 1990c]
Hartmann, G.; Drüe, S.: Verification of Continuity, Using Temporal Code. In: Proc. of the International Joint Conference on Neural Networks (IJCNN), II. San Diego 1990, pp. 459-464

[Hartmann 1991]
Hartmann, G.: Hierarchical Neural Representation by Synchronized Activity: A Concept for Visual Pattern Recognition. In: Taylor, J. G. et al. (ed.): Neural Network Dynamics. London et al. (Springer-Verlag) 1991, pp. 356-370

[Hartmann 1994]
Hartmann, G.; Drüe, S.; Dunker, J.; Kräuter, K. O.; Mertsching, B.; Seidenberg, E.: The SENROB Vision-System and its Philosophy. Accepted for Presentation at the Int. Conf. of Pattern Recognition, Jerusalem, October 1994

[Hubel 1959]
Hubel, D. H.; Wiesel, T. N.: Receptive Fields of Single Neurons in the Cat´s Striate Cortex. In: Journal of Physiology, 148, 1959, pp. 574-579