to appear in: Proc. of the Engineering Applications of Neural Networks
Conference EANN'96, London, 1996

Evaluation of local image structures applying a DCS network*

R. Herpers L. Witta[1] J. Bruske[2] G. Sommer[2]

GSF – Institut für Medizinische Informatik und Systemforschung, Medis
Ingolstädter Landstr. 1, 85764 Oberschleißheim, Germany
Email: herpers@gsf.de
[1] Current address: Lehrstuhl für Mensch-Maschine-Kommunikation, TU-München
Arcisstr. 12, 80290 München, Germany, Email: wta@mmk.e-technik.tu-muenchen.de
[2] Institut für Informatik, Christian-Albrechts-Universität
Preußerstr. 1-9, 24105 Kiel, Germany, Email: (jbr, gs)@informatik.uni-kiel.de

Abstract

In this contribution we present a face processing application based on **D**ynamic **C**ell **S**tructures (DCS) for classification of local image structures at particular keypoint positions. The DCS network is trained by supervised learning to recognize and classify feature vectors of several keypoints in face images such as the corners of the eyes or the intersections of the iris with the eyelid etc.. The keypoint positions are computed in advance by a model- and data-driven sequential search strategy applying a sophisticated filtering scheme. We apply the DCS network to classify detected image positions because of its ability to grasp the topological structure of complex input spaces as well as our good experiences with that type of network in number of other classification tasks. In our experiments the average error of false positive classifications is lower than 1% while the average error of false negative classifications is of about 5%. By combining the results of the model-driven localization and the DCS based data-driven verification the considered keypoints can be classified very reliably.

Keywords: Dynamic Cell Structures, RBF-Networks,
Computer Vision, Image Processing

1 Introduction

Detection and exact localization of certain characteristic keypoints in face images, such as the corners of the eyes or the mouth, are relevant issues for many applications in image processing. In general, these keypoints cannot be detected by purely data-driven methods. For example, the definition of an eye corner is more a semantic or high-level definition than a low-level one, i.e. it is not only based on the local image structure. Moreover, the local image structure of an eye corner and its local neighborhood shows large variations between different individuals, even between different images of the same individual.

In previous work we have developed a sequential search strategy which integrates appropriate model knowledge into a local data-driven processing strategy [4]. This algorithm is based on a very flexible and powerful filtering scheme [5] and consists of two steps:

1. Selection of the region of interest (ROI) using an attentive localization algorithm.
2. Detailed investigation of the selected local region applying a sequential search strategy to detect the included keypoints.

* *This work is partially supported by DFG grants So 320/1-1 and Ei 322/1-1.*

In the first step, motivated by the behavior of biological visual systems, the most salient regions of face images, such as the eye-, nose-, and mouth region, are localized applying a fast hierarchical algorithm based on an artificial attention mechanism [3].

In the second step the selected image regions are investigated using more sophisticated and expensive image processing methods. It is based on a sequential search strategy which integrates model knowledge about the considered scene to enable a robust and precise keypoint detection [4]. This sequential search strategy mainly uses the underlying line and edge information for the detection and tracking operations. The included domain knowledge allows a stepwise verification of the state of processing.

However, due to artifacts, occluded keypoints or unforeseen edge elements, the sequential search sometimes fails and stops at non keypoint positions. Therefore we have decided to augment our system with a third processing stage, a DCS based verification stage which is the topic of this article. Its task is to further decrease the error probability to take a non keypoint as the keypoint we are looking for. The combination of this verification stage with the sequential search strategy rests on Bayes' theorem.

We will introduce the neural verification stage and its combination with the sequential search in section 2. In section 3 we give the experimental results and in section 4 we draw a final conclusion.

2 The DCS based verification stage

In this section we briefly introduce DCS-networks together with some modifications developed for this application. Then we will explain the generation of feature vectors serving as input to our classifier and subsequently describe the design of the actual DCS based classifier. Finally we give the theoretical justification for the presented verification stage within the Bayesian framework.

2.1 Dynamic Cell Structures

Dynamic **C**ell **S**tructures (**DCS**) have been recently introduced in [1] and denote the class of RBF-based approximation schemes which attempt to concurrently learn and utilize **P**erfectly **T**opology preserving **F**eature **M**aps (**PTFM**s). DCS are a subclass of Martinetz's **T**opology **R**epresenting **N**etworks (**TRN**) [6] defined to contain any network using competitive Hebbian learning for building PTFMs.

The architectural characteristics of a DCS network are:

1. one hidden layer of radial basis functions (possibly growing/shrinking)
2. a dynamic lateral connection structure between these units and
3. one layer of (usually linear) output units.

The training algorithms for DCS adapt the lateral connection structure towards a PTFM by employing a competitive Hebbian learning rule and activate and adapt RBF units in the neighborhood of the current stimulus, where "neighborhood" relates to the simultaneously learned topology.

The **G**rowing **DCS** (**GDCS**) [1] is a DCS network which grows by inserting units according to a local error measure and the emerging PTFM. It is similar to Fritzke's GCS network [2] except in using a PTFM instead of a fixed hypertetrahedrical connection structure. GDCS thus addresses the major problems of Kohonen-type SOMS, i.e. the fixed number of neural units (GDCS incrementally grow and shrink), the fixed and usually not perfectly topology preserving neighborhood relation (DCS learn and utilize PTFMs instead) and a distribution of neural units that depends on the input probability distribution only (in GDCS the distribution of neural units also depends on the local error measure which can be chosen according to the task at hand). Finally, DCS networks can aid in automatic class/cluster separation by searching for connected components in the PTFM.

These properties of GDCS, its simplicity, efficiency and superior classification performance on a number of CMU classification benchmarks [1] made it the premier choice as the basis for our classifier. By learning PTFMs, DCS are able to grasp part of the topological structure of our high dimensional input space, and by exploiting this knowledge usually yield better classification results than simple nearest neighbor or other conventional classifiers.

For the application reported in this article we modified the Kohonen-type learning rule suggested in [1] for training the centers of the DCS neurons. One problem with this update rule is the correct choice of the learning factor for the best matching unit ϵ_{bmu} and for the direct neighbor unit ϵ_{nb} and the decreasing of ϵ_{bmu} and of ϵ_{nb} during the learning process. The **B**est **M**atching **U**nit **BMU** denotes the neural unit with highest activation on presentation of a feature vector $\vec{\nu}$ to the network:

$$a_i = e^{-\frac{1}{2\sigma^2} \quad \|\vec{\nu}-\vec{\mu_i}\|^2} \tag{1}$$

where a_i is the activation of unit i and $\vec{\mu_i}$ is its weighting vector.

We have modified the update rule by defining frequency modulated individual learning rates for each BMU $\epsilon_{bmu,i}$ and for each neighbor unit of the BMU $\epsilon_{nb,i}$. If the learning rate $\epsilon_{bmu,i}$ is substituted by $\frac{1}{n_{bmu,i}}$ and if $n_{bmu,i}$ is increased by 1 each time when neuron i is BMU, then the resulting learning rule for the BMU (first case of equation 2) computes the exact (non moving) average of all feature vectors which have fallen into the Voronoi polyhedron of neuron i since the beginning of the training:

$$\Delta\vec{\mu}_i = \begin{cases} \frac{1}{n_{bmu,i}} * (\vec{\nu} - \vec{\mu}_i) & \text{for } i = BMU, \\ \frac{1}{n_{nb,i}} * (\vec{\nu} - \vec{\mu}_i) & \text{for } i \in NB(BMU) \quad \text{and,} \\ 0, \quad \text{else.} \end{cases} \tag{2}$$

where $\vec{\mu}_i$ is the weight vector of neuron i and $\vec{\nu}$ is the feature vector during the training step. $NB(BMU)$ is the set of direct topological neighbors of the BMU.

Applying these individual learning rates the DCS is observed to rapidly converge during the first training periods. To avoid a "freezing" of the network caused by too small learning rates $\frac{1}{n_{bmu,i}}$, the increasing of $n_{bmu,i}$ is stopped if $n_{bmu,i}$ reaches a threshold calculated by $k_{bmu} \times z_{bmu,i}$, where k_{bmu} is a constant and $z_{bmu,i}$ counts the number of feature vectors that have fallen into the Voronoi polyhedron of neuron i during the training steps of the last training cycle. Therefore, after $k_{bmu} \times z_{bmu,i}$ feature vectors have fallen into the Voronoi polyhedron of neuron i the learning rate $\frac{1}{n_{bmu,i}}$ stops decreasing and the individual learning rule for the neuron i starts to behave in the same way like the original update rule (i.e. calculation of a moving average).

The learning rule for the neighbors of the BMU (second case of equation 2) is similar to the BMU case just presented. For the neighbor case also an individual learning rate $\frac{1}{n_{nb,i}}$ is held, where $n_{nb,i}$ counts the number of times when the neuron i has been a neighbor of a BMU. The factor k_{nb} restricts the growth of $n_{nb,i}$ and is chosen about 5 to 10 times as large as k_{bmu} .

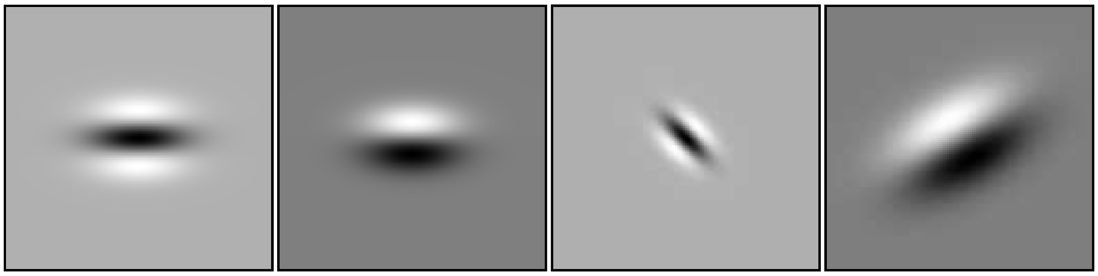
The fast convergence of the centers and the improved stability in regions of high data density support the rapid formation of PTFMs in these regions.

2.2 Feature generation

The feature vectors are obtained by applying a set of particular linear filters to the considered image position and its local 27×27 neighborhood. The projection coefficients of these filters are taken as components of our feature vectors. More specifically, we use the concept of steerable filters [5] to calculate many responses of different edge and line detectors (fig. 1) already utilized for the selection of the ROIs of the considered eye regions and for the sequential search strategy. Hence the projection coefficients have already been calculated.

Steerable filters were introduced to efficiently calculate the responses of filters in arbitrary orientations, scales, and other deformations [5]. The reconstruction of all deformed filters F_α is calculated by a superposition formula of the following type:

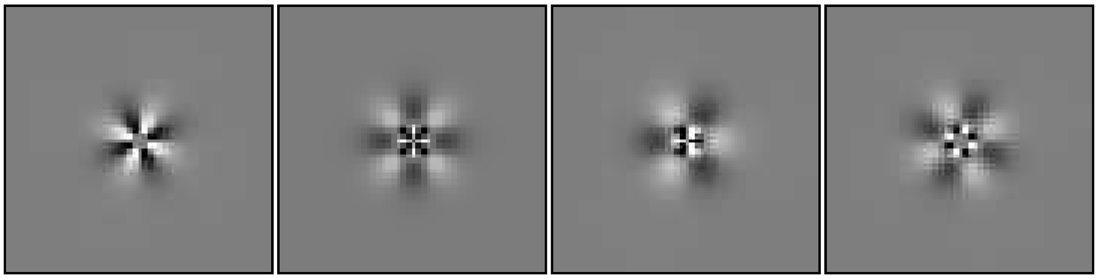
$$F_\alpha(\vec{x}) = \sum_{k=1}^{N} b_k(\alpha)\, A_k(\vec{x}) \tag{3}$$



(a) (b) (c) (d)

Figure 1: Second (a) and first (b) derivative of Gaussian filter for line and edge detection. These primitive filters are steered in orientation and scale (c,d). All rotated and scaled filters are computed from a small number of basis functions which optimally approximate any desired deformed filter.

Applying this technique it is possible to generate an infinite number of different 'deformed' filters (e.g. edge and line detectors steered in scale and orientation within a given interval) as a weighted sum of only a few so called **basis functions** (fig. 2). The number N of basis functions $A_k, k = 1 \ldots N$ is assumed to be small compared to the number of deformed filters. Typically N will be small (30 or 40 for example), while α can theoretically assume an infinite number of values and many thousands in practice (for orientation and scale). For example, 100 orientation steps and 17 distinct scales will result in 1700 different edge respectively line filters in a classical filtering scheme. With steerable filters, however, this huge number of filters is generated by only 30 (for the edge detector) respectively 40 (for the line detector) basis functions with an average approximation error of about 3% with respect to a kernel size of 27×27 pixels. For more details see [5].



(a) (b) (c) (d)

Figure 2: Examples of the basis functions to steer the line (a,b) and edge detection filter (c,d) in scale and orientation. Each considered image position is projected to 70 of these basis functions (30 for the edge detector and 40 for the line detector). The resulting projection coefficients form the components of the feature vector.

The projection coefficients of a considered image position within its 27×27 neighborhood to these basis functions establish our feature vector. The underlying assumption is that all the relevant and necessary line and edge information is encoded inside the 70 projection coefficients of the basis functions or 'weights' of the network. However, the generated representation is not complete, i. e. the representation is not sufficient to reconstruct an image area completely and error-free. Only the most important structural information is encoded and most of the non relevant information is lost. Furthermore, this representation causes an effective and useful data reduction of a factor of more than $10:1$ of a considered image area with a size of 27×27 pixels.

2.3 Class and activity map

In the presented application we have considered 9 different classes of keypoints with high morphological relevance: Inner eye corner, outer eye corner (for the left and the right eye each), (up to 4) intersections of the iris with the upper and lower eye lid and a point on the eye lid wrinkle. One has to remark that the intersections of the iris with the lower eye lid do not exist in every used eye region. Although most of the pixels in an eye region are not keypoints we do not introduce an explicit rest class, i.e. no class of non keypoint positions. The reason is that the network should not allocate neural units for a class we are not interested in. Instead we implicitly define the rest class by all pixels with an activity below a relative class dependent threshold. By presenting examples from the 9 keypoint classes, only the properties of the keypoints get represented by the network.

In order to generate the class map (fig. 3a), all pixels of the image region are classified by presenting their feature vectors to the DCS network. Since we have not explicitly defined a rest class, each pixel - even at non-keypoint positions - is labeled as belonging to a certain keypoint class (fig. 3a). Additionally, a second map, the activity map, is computed (fig. 3b) which encodes the activity induced by each pixel. The activity of a pixel is defined as the activity of the BMU on presentation of a feature vector $\vec{v}$ of the considered pixel position (equation 1). Combining the class map and the activity map we obtain the resulting classification map as depicted in (fig. 3c) where all the pixels with an activity below the class average have been assigned to the rest class.

Taking the 'left intersection of the iris boarder with the upper eye lid' keypoint class as an example, fig. 3d shows that only very few pixels belong to a particular keypoint class. Among them there are image positions in the direct neighborhood of the keypoint position and some pixels at positions which are totally unrelated to the original keypoint position. The latter pixels will contribute to the very low false positive decision probability (1% on average) (see section 2.4 ff). Obviously, this probability strongly depends on the selection of the class dependent activity threshold because the threshold is responsible for the assignment to the rest class.

(a)
(b)
(c)
(d)

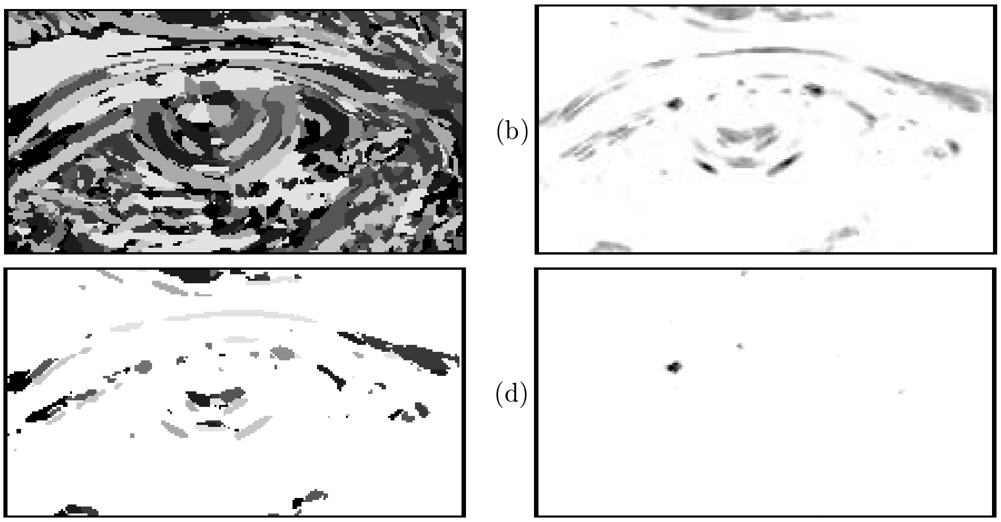


Figure 3: Classification results of the DCS network. (a) Class map for the entire considered image part without threshold; each pixel is assigned to a particular keypoint class encoded in distinct grey-levels. (b) Activity map of the classification result; high activity is encoded by dark grey-levels and low activity by bright ones. The activity is significantly increased, grey and dark regions are only visible at important or interesting image regions. (c) Resulting classification map of all keypoint classes after application of class dependent activity thresholds. All white pixels are assigned to the rest class and each grey-level encodes a distinct keypoint class. (d) Activity map related to the keypoint class 'intersection of the iris border with the upper left eye lid'; only very few image positions, at which false positive classifications can occur, can be found in the map.

2.4 Bayesian framework for verification

In a Bayesian framework the introduction of a verification stage for keypoint verification can be justified as follows: First, the sequential search for a keypoint of class C_i is modeled as a random experiment. The position at which the sequential search stops is assigned to the random variable X and hence $P(X = C_i)$ denotes the probability that the position we have stopped at really belongs to the keypoint class we have searched for. $P(X \neq C_i)$ is the error probability that the sequential search has failed, i.e. that we have stopped at a position not belonging to the desired keypoint class C_i.

In order to further reduce the error probability of taking a non keypoint as a keypoint of class C_i we introduce a second stage, the verification stage. This verification stage analyzes the position we have stopped at and classifies it as belonging to one of the n keypoint classes. Modeling the verification stage as a second random experiment and assigning the output of the classifier to the random variable Y, it decides for keypoint class C_i with conditional probability $P(Y = C_i|X = C_i)$ and $P(Y = C_i|X \neq C_i)$, respectively. The conditional classification error is determined by $P(Y = C_i|X \neq C_i)$ for false positive classification, and by $P(Y \neq C_i|X = C_i)$ for false negative classification. Now, for the combined system the a posteriori error probability of deciding for a keypoint of class C_i but not having stopped at such a keypoint is $P(X \neq C_i|Y = C_i)$. Using Bayes' theorem this probability calculates as

$$P(X \neq C_i|Y = C_i) = \frac{P(Y = C_i|X \neq C_i)\ P(X \neq C_i)}{P(Y = C_i)} \tag{4}$$

$$= \frac{P(Y = C_i|X \neq C_i)\ P(X \neq C_i)}{P(Y = C_i|X \neq C_i)\ P(X \neq C_i) + P(Y = C_i|X = C_i)\ P(X = C_i)} \tag{5}$$

Obviously, the verification stage does only make sense if the a posteriori error probabilities $P(X \neq C_i|Y = C_i)$ turn out to be smaller than the a priori probabilities $P(X \neq C_i)$. In section 3 we will show that due to the small conditional error probabilities of the DCS based verification stage they are indeed *much* smaller.

3 Results

3.1 Verification of keypoints with the DCS

For training the network we use feature vectors calculated from keypoint positions and their near neighborhood as mentioned before. The overall number of examples was more than 660 which was obtained from 95 different images of left and right eyes. Feature vectors from keypoints of 75 of the 98 eye images built the training set. The remaining 23 eye regions contributed to the test set. The false negative classification error averaged over all keypoint classes on the training and the test set as a function of the number of neural units inserted in the DCS network is shown in (table 1). Since the (false negative) classification error on the test set does not fall below 5.6% for more than 120 neural units this is the point to stop the growing of the DCS network. Otherwise generalization abilities can be expected to decrease.

Table 2 shows the results for the combined search and verification system for four joined keypoint classes. In this table, $\hat{P}(X \neq C_i)$ denote the empirical prior error probabilities of the sequential search strategy, $\hat{P}(Y = C_i|X \neq C_i)$ the empirical conditional probabilities for false positive classification, $\hat{P}(Y \neq C_i|X = C_i)$ the empirical conditional probabilities for false negative classification and, finally, $\hat{P}(X \neq C_i|Y = C_i)$ the empirical a posteriori error probabilities of the combined system. The latter are much lower than the priors, due to the low conditional classification error of the DCS.

3.2 Detection of keypoints with the DCS

In an additional experiment we tried to directly detect the keypoints with the DCS, i.e. without the sequential search strategy. For this end we generate the class map and search the pixel with maximum activity for each keypoint class.

Network size						
neurons	21	34	60	90	120	142
Training set (527 pattern):						
mistakes	52	41	31	19	7	0
error(%)	9.8	7.7	5.8	3.6	1.3	0
Test set (141 pattern):						
mistakes	11	10	10	9	8	8
error(%)	7.8	7.0	7.0	6.3	5.6	5.6

Table 1: Classification error related to the size of the DCS network

Keypoint	$\hat{P}(X \neq C_i)$	$\hat{P}(Y = C_i \mid X \neq C_i)$	$\hat{P}(Y \neq C_i \mid X = C_i)$	$\hat{P}(X \neq C_i \mid Y = C_i)$
inner eye corner	19.0	1.5	0.0	0.0
outer eye corner	12.5	1.9	4.8	0.0
intersec. iris/eye lids	19.4	0.7	5.4	1.4
eye lid wrinkle	13.3	1.1	0.0	0.0
overall	17.0	1.1	3.8	0.8

Table 2: Empirical error probabilities in %

The computation of the class map (fig. 3a) and the activity map (fig. 3b) for a typical 230×135 image part takes about 28 sec. on a SGI-Indy R4400 workstation. The results for this purely data-driven keypoint detection are summarized in table 3.

	inner eye corner	outer eye corner	intersec. iris/lids	eye lid wrinkle
error (%)	56.5	47.8	11.9	61.9

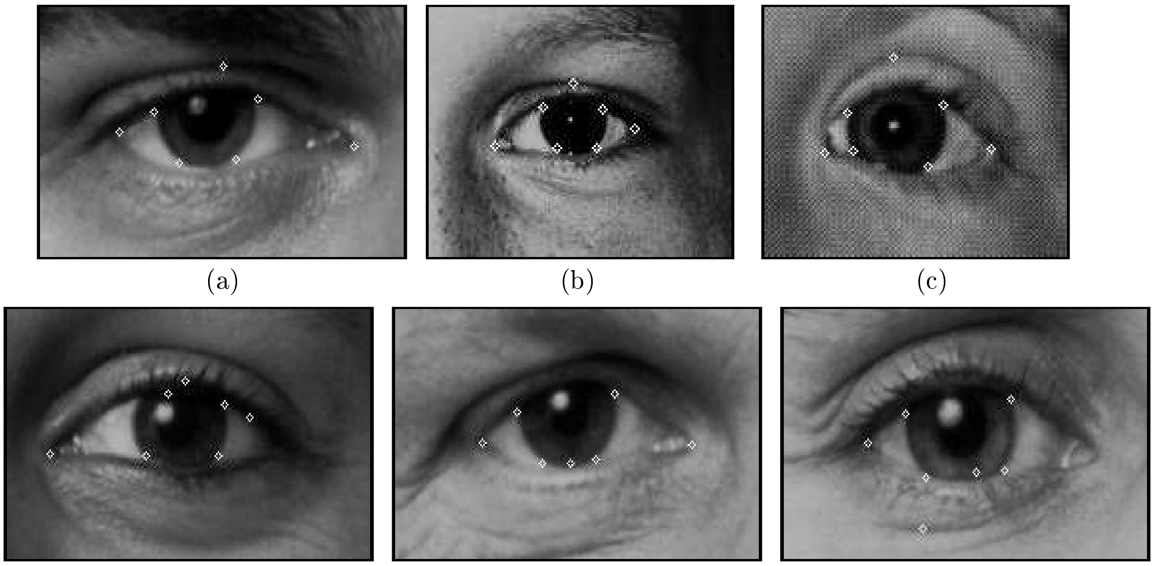
Table 3: Purely data-driven detection error

The relatively high error rate for the keypoint detection respectively the detection of the eye corner keypoints and the eye lid keypoint is not due to the DCS network failing to assign the correct class to keypoint positions but rather to other image structures somewhere in the eye region inducing more activation in the network than the exact keypoint position. The reason is that in complex images a lot of image structures exist which have a comparable or very similar image structure as the exact keypoint and that, by training the network on keypoints only, we did not teach the net to discriminate between keypoints and these similar structures.

For illustration, the keypoints detected by the purely data-driven keypoint search are depicted in fig. 4. The 3 images in the upper line are taken from the training set and the 3 images in the lower line from the test set. In the image parts of test set the keypoints of the eye lid wrinkle and of the eye corner are sometimes misclassified. The class maps respectively the activity maps depicted in fig. 3 belong to the eye region of fig. 4a.

4 Conclusion

In this paper we have presented the application of a DCS network for the verification of keypoint positions in face images which have been computed in advance by a model- and data-driven sequential search strategy. The DCS network has been demonstrated to classify keypoint locations suggested by the sequential search strategy with high reliability and the results for the combined search and classification scheme as expressed in the last column of (tab. 2) are very promising. The detection results depicted in table 3 indicate that purely data-driven methods are unlikely to lead to satisfactory results for keypoint detection in complex high resolution face images although



(a) (b) (c)

(d) (e) (f)

Figure 4: Examples of detected keypoints in 3 eye regions of the training set (a-c) and 3 eye regions of the test set (d-f). The depicted results are computed by direct classification of each image position of an eye region with a well trained DCS network but without sequential search of the keypoint in advance. Since the network has been trained with keypoints only (no negative examples) not all keypoints of the test set can be classified correctly. In particular, the keypoint on the eye lid wrinkle is often misclassified. The keypoints of the intersection points of the iris with the eye lid, however, are detected reliably without any preliminary sequential search because of the very characteristic image structure.

performance could be improved by a modified training scheme. The presented hybrid approach of a model-based search strategy followed by a data-driven neural network verification component outperforms each of its constituents taken for its own. It is an example of how well engineered methods for image analysis can benefit from additional state of the art neural network based components.

Acknowledgements

We are grateful to Dr. M. Michaelis from the Medis-Institute at the GSF Research Center for providing the steerability scheme and for many helpful discussions. The work has been partially supported by DFG grants So 320/1-1 and Ei 322/1-1.

References

[1] J. Bruske and G. Sommer, *Dynamic cell structure learns perfectly topology preserving map*, Neural Computing, Vol. 7, no. 4, pp. 845-865, 1995.

[2] B. Fritzke, *Growing cell structures - a self organizing network for unsupervised and supervised training*, ICSI Berkeley, Tech.-Rep., tr-93-026, 1993.

[3] R. Herpers, H. Kattner, H. Rodax, and G. Sommer, "GAZE: An attentional processing strategy to detect and analyze the prominent facial regions", *Proc. of the Int. Workshop on Autom. Face- and Gesture-Rec.*, M. Bichsel (ed.), Zurich, Switzerland, pp. 214-220, 1995.

[4] R. Herpers, M. Michaelis, G. Sommer, and L. Witta, *Detection of keypoints in face images*, Tech.-Rep. GSF-Bericht 24/95, Sept. 1995, D-85764 Oberschleissheim, Germany, 1995.

[5] M. Michaelis, *Low level image processing using steerable filters*, PhD thesis, Christian-Albrechts-Universität, D-24105 Kiel, Germany, 1995.

[6] T. Martinetz, *Competitive Hebbian learning rule forms perfectly topology preserving maps*, Proc. ICANN 93, pp. 426-438, 1993.